CP/M

OPERÁCIÓS RENDSZER

LSI Alkalmazástechnikai Tanácsadó Szolgálat

CP/M

OPERÁCIÓS RENDSZER

Szerkesztette: Szenes Katalin

Lektorálta:

Beszeda Tamás
Szenes Katalin
Téglás György

Készitette:

Erdős Iván
Fazekas György
Lipovszky György
Subai László
Szenes Katalin
Téglás György
Unger Anna
Vida Tibor

LSI ALKALMAZÁSTECHNIKAI TANÁCSADÓ SZOLGÁLAT

Budapest, 1985.

„A MIKROSZÁMÍTÓGÉPEK ÉS ALKALMAZÁSI RENDSZEREIK KUTATÁSA–FEJLESZTÉSE"

című

OMFB TÁRCAPROGRAM

4/b. Alprogramjához kapcsolódóan készült

Kézikönyv

OMIKK
LSI Alkalmazástechnikai Tanácsadó Szolgálat
Telefon: 805–712

Témafelelős:
SZIKLAI KLÁRA

ISBN 963 592 349 X
963 592 351 1

Utánnyomás

Készült az OMIKK házinyomdájában
Budapest, I., Gyorskocsi u. 5–7.
Felelős vezető: Tóth Károly
Engedélyszám: 46 254
Táskaszám: 1014

T A R T A L O M

Oldal

ELŐSZÓ

I.RÉSZ: AMIG A CP/M BEJELENTKEZIK 1

1. BEVEZETÉS 2

2. MI A SZEMÉLYI SZÁMITÓGÉP 4

3. ISMERKEDÉS A CP/M-MEL 8

3.1. COMMODORE 64 1Ø

3.2. MØ8X és PROPER-8 12

II.RÉSZ: CP/M ALAPISMERETEK 14

1. MEMÓRIAFELOSZTÁS 15

2. A CP/M RÉSZEI ÉS BELSŐ SZERKEZETE 17

3. ÁLTALÁNOS TUDNIVALÓK A LEMEZRŐL 2Ø

5. ALAPVETŐ REZIDENS PARANCSOK 33

5.1. DIR /directory/ 33

5.2. TYPE 35

5.3. REN /rename/ 35

5.4. ERA /erase/ 36

5.5. SAVE 37

6. ALAPVETŐ TRANZIENS PARANCSOK 39

6.1. SYSGEN 39

6.2. STAT 4Ø

6.3. SUBMIT 51

SUBMIT XSUB-bal 54

6.4. PIP 56

6.5. ED 58
6.6. ASM 59
6.7. DDT 59
6.8. LOAD 6Ø
6.9. DUMP 6Ø
6.1Ø. MOVCPM 61

III.RÉSZ: FILE-KEZELÉS, SZERKESZTÉS, PROGRAMOZÁS CP/M ALATT 62

1. FILE-KEZELÉS CP/M ALATT: PIP 63
1.1. PIP-funkciók 63
1.2. PIP-kulcsszavak 71
1.3. A PIP-parancsokban alkalmazható paraméterek 73
1.4. A PIP uj lehetőségei a 2.2 verzióban 81
Másolás felhasználói területekről 82
Csak olvasható file-ok 84

2. SZERKESZTÉS CP/M ALATT: AZ EDITOR /ED/ 85

2.1. Bevezetés 85
2.2. Belépés az ED-be 86
2.3. Az editor parancsairól általában 88
2.4. Az ED müködése 89
2.5. Forrásszöveg mozgatása az editorral 91
2.6. Szöveg beszurás 92
2.7. A szerkesztés befejezése 93
2.8. Elemi szerkesztési parancsok 94
2.9. Az editor további lehetőségei 96
2.9.1. Szöveg keresés és módositás 96
2.9.2. Forráskönyvtárak 99
2.9.3. Parancsok ismételt végrehajtása 1ØØ
2.9.4. Emelt és alaphelyzetü billentyüzet 1Ø1
2.9.5. Utasitáspozicionálás 1ØI

2.9.6. Sorszámozás 1Ø2
2.1Ø. Az ED hibaüzenetei 1Ø2
2.11. Az editor-parancsok összefoglalása 1Ø3

3. AZ ASSEMBLY NYELVÜ PROGRAMOZÁS TÁMOGATÁSA A CP/M OPERÁCIÓS RENDSZERBEN 1Ø5

3.1. Bevezetés 1Ø5
3.2. CP/M assembler: ASM 1Ø6
3.1.1. Bevezetés 1Ø6
3.2.2. Az assembly forrásnyelvi programjának formája 111
3.2.3. Az operandusok formája 113
3.2.3.1. Cimkék 113
3.2.3.2. Numerikus konstansok 114
3.2.3.3. Fenntartott szavak 116
3.2.3.4. Szövegkonstansok 118
3.2.3.5. Aritmetikai és logikai utasitások 118
3.2.3.6. Az utasitások sorrendje 12Ø
3.2.4. Assembler direktivák 121
3.2.4.1. Az ORG direktiva 122
3.2.4.2. Az END direktiva 123
3.2.4.3. Az EQU direktiva 124
3.2.4.4. A SET direktiva 125
3.2.4.5. Az IF és ENDIF direktivák 125
3.2.4.6. A DB direktiva 126
3.2.4.7. A DW direktiva 127
3.2.4.8. A DS direktiva 128
3.2.5. Utasitás elnevezések 129
3.2.5.1. Ugró utasitások, szubrutinhivások, visszatérő utasitások 13Ø
3.2.5.2. Müveletek konstansokkal /Immediate Instructions/ 132

3.2.5.3. Regiszterek értékének növelése és csökkentése /Increment, decrement instructions/ 133
3.2.5.4. Adatmozgató utasitások 134
3.2.5.5. Aritmetikai és logikai müveletek 135
3.2.5.6. Vezérlő utasitások 137
3.2.6. Hibaüzenetek 138

3.3. Dinamikus hibakereső program: DDT /Dynamic Debugging Tool/ 14Ø
3.3.1. Bevezetés 14Ø
3.3.2. DDT parancsok 143
3.3.2.1. Az A /Assemble/ parancs 143
3.3.2.2. A D /Display/ parancs 144
3.3.2.3. Az F /Fill/ parancs 145
3.3.2.4. A G /Go/ parancs 146
3.3.2.5. Az I/Input/ parancs 147
3.3.2.6. Az L /List/ parancs 148
3.3.2.7. Az M /Move/ parancs 148
3.3.2.8. Az R /Read/ parancs 149
3.3.2.9. Az S /Set/ parancs 15Ø
3.3.2.1O. A T /Trace/ parancs 151
3.3.2.11. Az U /Untrace/ parancs 153
3.3.2.12. Az X /Examine/ parancs 153

3.4. A hexadecimális programokat betöltő LOAD program 154
3.5. A memória listázása: DUMP program 156

IV.RÉSZ: A CP/M PARANCSOK ÉS RENDSZERPOGRAMOK ÖSSZEFOGLALÁSA 158

1. ASM 16Ø

2. DDT 163

3. DIR 164

4. DUMP 166

5. ED 167

6. ERA 168

7. LOAD 17Ø

8. MOVCPM 171

9. PIP 173

1Ø. REN 178

11. SAVE 179

12. STAT 18Ø

13. SUBMIT 183

14. SYSGEN 185

15. TYPE 187

16. USER 188

17. XSUB 189

V. RÉSZ: A CP/M OPERÁCIÓS RENDSZER BELÜLRŐL 191

1. A CP/M RENDSZER FELÉPITÉSE 192

1.1. A CP/M modulok kapcsolata 194

2. A CP/M BELSŐ VILÁGA 197

2.1. A CP/M operációs rendszer betöltése 198

2.2. A BIOS ugrótábla 199

2.3. Az IOBYTE 2Ø1

2.4. A BIOS szubrutinok részletes leirása 2Ø4

2.5. A Ø. lap memóriafelosztása 211

2.6. A lemezegységhez tartozó paraméter tábla:
DPT /Disc Parameter Table/ 213

2.6.1. A DPT fejléc 213
2.6.2. A DPT fejlécelemek alkotórészei 215
2.7. A DISKDEF /disc definiáló/ makrókönyvtár használata 22Ø

3. RENDSZERFUNKCIÓK 226

3.1. Rendszerfunkcióju szubrutinok 232
Ø. rutin: Rendszer inicializálás 232
1. rutin: Kezelői klaviatura bemenet 233
2. rutin: Kezelői display kimenet 234
3. rutin: Olvasói bemenet 234
4. rutin: Lyukszalag lyukasztói kimenet 235
5. rutin: Listázási kimenet 235
6. rutin: Közvetlen kezelői bemenet/kimenet 236
7. rutin: Add meg az IOBYTE-ot 237
8. rutin: Állitsd be az IOBYTE-ot 237
9. rutin: Irj ki egy szöveget 237
10. rutin: Olvasd az operátori input-puffert 238
11. rutin: Add meg a kezelői klaviatura állapotát 239
12. rutin: A verziószám megadása 239
13. rutin: A lemezegység inicializálása 24Ø
14. rutin: A lemezegység kiválasztása 241
15. rutin: Nyisd meg a file-t 241
16. rutin: Zárd be a file-t 243
17. rutin: Keresd az elsőt 244
18. rutin: Keresd a következőt 245
19. rutin: Töröld ki a file-t 245
20. rutin: Soros olvasás 246
21. rutin: Soros irás 247
22. rutin: Készits file-t 248
23. rutin: A file nevének megváltoztatása 249
24. rutin: A lemezegység müködésének állapota 25Ø

25. rutin: A kurrens lemezegység neve 25Ø
26. rutin: Állitsd be a DMA cimet 251
27. rutin: A lemezegység információs vektorának cime 252
28. rutin: Lemezegység irásvédelme 253
29. rutin: Add meg a csak olvasható lemezegységeket 253
3Ø. rutin: Állitsd be a file-információkat 254
31. rutin: A lemezegység paraméterei cimének beolvasása 255
32. rutin: A felhasználó kód beállitása/lekérdezése 256
33. rutin: Közvetlen olvasás 257
34. rutin: Közvetlen irás 259
35. rutin: A file méretének számitása 259
36. rutin: Állits be közvetlen rekordelérést 26Ø

Rendszerfunkcióju rutinok összefoglaló táblázata 261

4. RENDSZERGENERÁLÁS: A CP/M INSTALLÁLÁSA ÉS MÓDOSITÁSA 264

4.1. Elsőszintü rendszergenerálás: GETSYS, PUTSYS 266

4.2. Második szintü rendszergenerálás: MOVCPM 269

F Ü G G E L É K E K 274

I. CP/M PARANCSOK 275

II. AZ EDITOR UTASITÁSAI 281

III. LOGIKAI ÉS FIZIKAI ESZKÖZNEVEK 288

IV. PIP KULCSSZAVAK ÉS PARAMÉTEREK 289

V. KÉPERNYŐSZERKESZTÉS: VEZÉRLŐ /KONTROL/ KARAKTEREK 294

VI. A CP/M FONTOSABB HIBAÜZENETEI 295

VII. A CP/M ÁLTAL TÁMOGATOTT FILE-TIPUSOK 299

VIII. CP/M - KOMPATIBILIS PROGRAMOK 301

IX. DDT MINTAPÉLDA 309

X. BIOS UGRÓTÁBLA FELÉPITÉSE 313

XI. DPT ÉS DPB 2-LEMEZES RENDSZER ESETÉN 314

XII. BIOS MINTAPÉLDA 316

XIII. COPY MINTAPROGRAM 322

XIV. GETSYS ÉS PUTSYS MINTAPROGRAMOK 329

A kézikönyvben alkalmazott jelölések 332

A kézikönyvben szereplő röviditések jegyzéke 334

INDEX 336

E L Ő S Z Ó

A CP/M a legelterjedtebb mikroszámitógépes operációs rendszer. Az általánosan alkalmazható rendszer-, és felhasználói programokat mind nyugaton, mind hazánkban igen sokan készitik ugy, hogy CP/M "alatt" futtassanak.

Átfogó, a rendszert felhasználói és rendszerprogramozói szempontból is ismertető magyar nyelvü leirás eddig még nem készült; most ezt a hiányt kivánjuk pótolni.

E könyvet egyaránt ajánljuk azoknak, akik még csak most ismerkednek a mikroszámitógépekkel, azoknak, akik BASIC, assembler vagy valamely más nyelv ismerete és használata révén már valamilyen szinten kapcsolatba kerültek e gépekkel,és azoknak is, akik már használják a CP/M-et, de a "belsejére" is kiváncsiak lennének vagy esetleg saját tervezésü gépükön kivánják megvalósitani. A könyv szerkezete is tükrözi ezt a célkitüzést: az egyszerütől a bonyolult felé haladva mind részletesebb információval szolgálnak az egyes fejezetek.

A kezdőknek bemutatjuk, hogy néhány Magyarországon elterjedt géptipuson hogyan lehet "megszólaltatni" a rendszert, majd ismertetjük azokat a legfontosabb lehetőségeket, melyek ismeretében már el lehet indulni.

Külön-külön fejezeteket szántunk az információ perifériák közötti, illetve a processzor és a perifériák közötti mozgatási lehetőségeinek, a program és adat file-ok szerkesztésének,továbbá annak leirására, hogyan támogatja a CP/M az assembly nyelvü programozást. Ez utóbbi fejezet keretében, a szemléletesség kedvéért meglehetős részleteséggel ismertetjük a CP/M alatt leggyakrabban használt assemblert, az Intel 8Ø8Ø-at. Ezt, az adott gép assembler forditóprogramjának leirását, és egy Z8Ø mnemoniktáblát felhasználva, ha a gép "tudja" a Z8Ø lehetőségeit, az olvasó Z8Ø assemblerben is képes lesz programozni.

A gyors tájékozódást megkönnyitendő, ezután röviden összefoglaljuk a rendszerparancsokat abc-sorrendben is.

Ezután térünk rá a CP/M belső felépitésének ismertetésére, melyen belül leirjuk a CP/M memóriatérképét, részeit és azok kapcsolatait, az assemblerben hivható rendszerfunkciókat, végül a rendszergenerálás folyamatát.

Az operációs rendszerrel kapcsolatos legfontosabb információkat függelékek formájában is hozza a könyv; szerepel ezek között a rendszerparancsok összefoglaló táblázata, az editor parancsainak felsorolása, a CP/M által használt logikai és fizikai eszköznevek, a PIP kulcsszavak és paraméterek összefoglalása, a képernyőszerkesztést végző vezérlőkarakterek ismertetése, a rendszer hibaüzeneteinek leirása, a CP/M alatt futtatható programok körének vázlatos ismertetése, továbbá

néhány olyan mintapélda, mely a rendszerprogramozók számára megkönnyiti a könyvben leirtak jobb megértését.

A kézikönyv összeállitásában részt vevő munkatársak többféle géptipuson dolgoztak; szerepelt ezek között pl. az ICC /Intelligent CAMAC Controller/, az MØ8X, az ABC 8Ø, a Commodore 64 és a VT2Ø/A. Információink azonban, hála a CP/M szerkezetének, amely lehetővé teszi, hogy a hardware architekturák eltérése ellenére a rendszer, a hardware-hez kötődő rutinok kivételével, lényegében ugyanaz legyen, nem ellentmondásosak. Az egyes verziók közötti eltérésekre, ahol ez lényeges volt, külön felhivtuk a figyelmet.

Számitástechnikai szakszövegben mindig problematikus a terminológia; ezt a problémát, ha megoldani nem is próbáltuk, ugy kezeltük, hogy lehetőleg egységesen és következetesen ugyanazokat a kifejezéseket használtuk ugyanazon fogalmak leirására. Ahol szükségesnek látszott, inkább többször elmagyaráztuk a fogalmak jelentését /pl. blokk-file-terület/.

Reméljük, hogy a könyvet haszonnal forgatják majd, mindazonáltal köszönettel veszünk a könyvvel kapcsolatos mindennemü észrevételt és helyreigazitást; szükség esetén ezeket közzé is tesszük.

Szenes Katalin

I. RÉSZ:

AMIG A CP/M BEJELENTKEZIK

1. B E V E Z E T É S

A CP/M operációs rendszert Gary Kildall definiálta, s hozta létre első változatát. Kildall abban az időben /1973/ az Intelnél dolgozott és a PL/M nyelv rezidens forditóprogramjának elkészitésében müködött közre. Az első mikroszámitógépek input/output eszköztár csupán a lyukszalagra korlátozódott, amely teljesen alkalmatlan volt komolyabb feladatok megoldására, igy Kildall figyelme az éppen kifejlesztés alatt álló hajlékony lemez /floppy-disc/ felé fordult. Barátjával, John Torodedal hamar rájöttek, hogy a floppy csak akkor szolgálhat háttértárolóként, ha azt megfelelő hardware-software környezettel látják el. Igy Kildall file-kezelőt /a CP/M BDOS előfutárát/, Torode lemez-vezérlőt készitett, s lényegében a kettő együtt már a CP/M legelső verzióját jelentette. A kezdeti igen csekély érdeklődés után 1975-ben kezdődött a CP/M széleskörü elterjedése. Az első széles körü alkalmazó az azóta feloszlott IMSAI cég, az alkalmazott verzió a CP/M 1.3, amely már tartalmazta az assemblert /ASM/, a debuggert /DDT/ és az editort /ED/ is. Az ezt követő legjelentősebb módositást a 2.2. verzió megjelenése jelentette, amely már az időközben megszületett nagy kapacitásu lemeztárolók lehetőségeinek kihasználására készült.

Mi okozza a CP/M nagy népszerüségét, hiszen ma a mikroszámitástechnikában messze ez a legszélesebb körben alkalmazott és elfogadott operációs rendszer? Elsősorban az, hogy nem kötődik egyetlen gyártó hardware-konfigurációjához, hanem pontosan definiált módositásokkal és

kiegészitésekkel csaknem valamennyi mikroszámitógépen futtatható. Ez nagy előrelépés volt, hiszen igy a rendszer- és egyéb programok /BASIC interpreterek, szövegszerkesztők, stb./ változtatás nélkül alkalmasak más gépeken való felhasználásra is, mint amelyen kifejlesztették azokat. A CP/M létrejötte óta igen nagyszámu programot dolgoztak ki az operációs rendszer felügyelete alá, a legkülönbözőbb alkalmazásokra. Több, mint 1ØØ cég kinál ilyen programtermékeket és a számitógép-hobby klubok, magazinok, könyvek körében is mind népszerübbek a CP/M-kompatibilis termékek. Bár a CP/M-et sok kritika éri azok részéről, akik több szolgáltatást nyujtó speciális operációs rendszerekhez szoktak, a CP/M mégis gyakorlatilag a mikroszámitógép-alkalmazók szabványos programfejlesztési környezetévé vált.

A CP/M további fejlesztése egyértelmüen a hálózati felhasználások támogatásának irányába mutat. Az MP/M és a CP/Net egyaránt olyan termékek, amelyek megtartva a kompatibilitást a korábbi CP/M operációs rendszerekkel, lehetővé teszik mikroszámitógépek hálózati együttmüködését, ellentétben az eddigi szigoruan egyfelhasználós rendszerekkel. A mikrogépek alkalmazása a hálózatban szereplő nagyméretü /nagygépes/ adatbázisok elérésére szintén sokakat érdeklő probléma. A CP/M továbbfejlesztésektől remélhető, hogy a közeli években többé-kevésbé szabványos megoldást nyerhetnek a fenti feladatok.

2. MI A SZEMÉLYI SZÁMITÓGÉP?

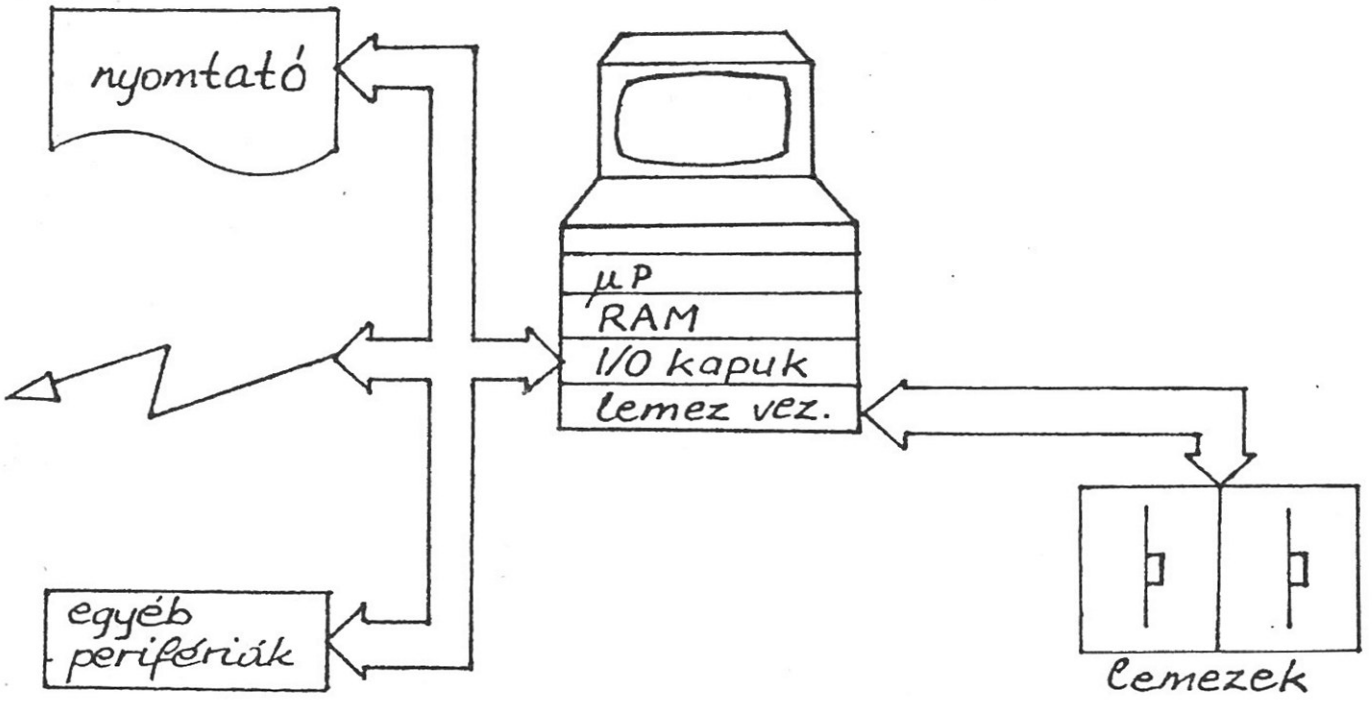
Az 1970-ben megjelent mikroprocesszor mindinkább kezdi átalakitani világunkat. Egyre több hétköznapi használati tárgyunkba "lopódzik be" és több-kevesebb "intelligenciát" kölcsönöz neki, igy személyautókba, konyhai sütőkbe, mosógépekbe kerül beépitésre. Emellett az utóbbi néhány évben, kb. 1975 óta megjelentek a piacon azok a kisméretü, de meglepően nagy teljesitményü eszközök, az un. személyi számitógépek, amelyek elektronikus "agya" szintén mikroprocesszor.

Hogyan alkalmazható a mikroprocesszor ilyen sok célra? A válasz a mikroprocesszor igen hajlékony és hatékony mikroprogramozhatóságában rejlik, amely lehetővé teszi igen eltérő alkalmazások támogatását is, mint a mindennapi élet példái bizonyitják.

Mint általában a bonyolult feladatok, két különböző orientációju személyi számitógép-alkalmazás mindig többé-kevésbé különbözik egymástól. Mégis vannak olyan programelemek, amelyek szinte valamennyi alkalmazásban szükségesek és csaknem változtatás nélkül megtalálhatók. Ezeket az elemeket érdemes összegyüjteni és egyszer általános formában megirni, majd a változatos programozási feladatokban változtatás nélkül felhasználni; ezek összességét nevezzük operációs rendszernek. Ilyen közös feladatok pl. a billentyüzet, a képernyő, a lemez kezelése, a programok felügyelete. Ha ezek az elemek készen rendelkezésre állnak, a programozó a feladat specifikus részeire koncentrálhat, nem

kell sokat foglalkozni a rutinszerüen ismétlődő részfeladatokkal.

Mielőtt azonban megismerkednénk az operációs rendszer feladataival és müködésével, néhány szót kell ejteni magáról a számitógéprõl is. A ma forgalomban és használatban lévõ számitógépeket három kategóriába szokás csoportositani: nagyszámitógépekre /mainframes/, miniszámitógépekre és mikro- vagy személyi számitógépekre. A hardware elemek fantasztikus fejlődésével és miniatürizálódásával a különbség az egyes kategóriák között mind méretben, mind kapacitásban és sebességben csökken, igy az ujabb gépek besorolása a fenti kategóriákban bonyolult, ha éppen nem lehetetlen. A CP/M-mel kapcsolatos ismertetésünk szempontjából a mikroszámitógép egy 8-bites mikroprocesszor köré épitett rendszer, amelynek komponensei az I-1. ábrán láthatók.



I-1. ábra

A központi egység általában a billentyüzettel és nemritkán a képernyős kijelzővel /más néven: video monitor, display, terminál/ is egybe van épitve. A központi egység két fő része a mikroprocesszor maga /CPU-central processing unit/ és a központi tár /RAM - random access memory : irásra és olvasásra egyaránt használható memória és ROM - read only memory: csak olvasható memória./. Az már a konkrét számitógéptipustól függ, hogy a lemezvezérlő egység, valamint a többi perifériák vezérlői a központi egységben, vagy magukban a perifériákban kapnak-e helyet, de ez a programozó számára többé-kevésbé közömbös is. Ha CP/M-et akarunk használni, mindenképpen szükségünk van egy- vagy két egységből álló hajlékony lemezes tárolóra. Ezeken az alapvető perifériákon kivül opcionálisan tartalmazhat rendszerünk sornyomtatót, plottert, a távoli kapcsolatot lehetővé tevő modemet és sok más speciális perifériát is. Utóbbiak közül mind nagyobb körben válnak elérhetővé a nagy kapacitásu fixen beépitett lemezes /un. Winchester/ tárolók is, amelyek óriásira /akár többszáz Mbyte-ra/ növelhetik gépünk tárolóképességét.

A fent körvonalazott számitógép ugyan nagy teljesitményü, változatos feladatok megoldására alkalmas eszköz, azonban ahhoz, hogy ez könnyen megvalósitható legyen, nem elégségesek a hardware nyujtotta lehetőségek, általánosan használható programeszközökre, software-termékekre is szükségünk van. Ezek az eszközök tulajdonképpen három nagy csoportra oszthatók: nyelvi eszközökre, operációs rendszerre és rendszer/segéd/-programokra /utilitykre/, bár szétválasztásuk és besorolásuk az egyes csoportokba nem mindig könnyü feladat.

A nyelvi eszközök lehetővé teszik, hogy a gépeket ne a valamennyi lehetőséget nyujtó/de igen bonyolultan és nagy munkaráforditással alkalmazható/ gépi kódban programozzuk, hanem konkrét feladatunk ismeretében kiválaszszuk a korlátozást nem jelentő, de a feladatmegoldást igen megkönnyitő programozási nyelvet. E programozási nyelveket forditó-programok /compiler-ek/ vagy interpreterek segitségével használhatjuk. Az előbbiek a gép által közvetlenül értelmezhető utasitássorozatokra forditják le a felhasználó programutasitásait, az utóbbiak utasitásról utasitásra értelmezik és végrehajtják azt, ennek következtében általában lassabbak, de a nem tul nagy tapasztalattal, szakképzettséggel rendelkező felhasználó egyszerübb feladatait gyorsan megoldhatóvá teszik. Utóbbira példa lehet a személyi számitógépek szinte kötelező BASIC-interpretere, mig az előbbiekre a gépikódu utasitásokat mnemonikokkal /emlékeztető-nevekkel/ helyettesitő és a cimzésmódokat intelligensebb módon kezelő assemblerek, amelyek viszont a rendszerfejlesztés legszélesebb körben elterjedt /bár nem egyedülálló!/ eszközei. Az utóbbi időben a mikroszámitógépek egyre bővülő lehetőségei megengedték, hogy azokon magasszintü nyelveket implementáljanak, mind olyanokat, amelyek már nagygépes környezetből jólismertek /mint pl. FORTRAN, PASCAL/, ill. olyanokat, amelyeket többé-kevésbé a mikroszámitógépek tulajdonságait és követelményeit figyelembe véve alakitottak ki /pl. FORTH, C, PL/M /. Ezek közül némelyek a rendszerfejlesztés intelligens eszközeiként is beváltak.

Az operációs rendszer főbb feladatait már futólag érintettük. Ezek elsősorban a perifériák, kiemelten a lemez kezelése, file-kezelés, a gépen való munka környezetének

biztositása. Ma már nagyon nehéz pontosan és ugyanakkor általánosan definiálni az operációs rendszer, a nyelvi rendszer és a rendszersegédprogramok határait. Igy pl. a szövegszerkesztő programok általában a rendszersegédprogramok közé sorolhatók, de a dedikált /a nyelvi elemeket ismerő/ nyelvi szerkesztők egyértelmüen a nyelvi rendszerhez tartoznak, mig pl. a CP/M esetében az editort és még egy néhány egyéb programot a tranziens parancsok, tehát az operációs rendszer részeként szokás tárgyalni.

A rendszersegédprogramok /utility-k/ közé olyan programtermékek tartoznak, amelyek a programfejlesztés, - alkalmazás lebonyolitásában segitséget nyujtanak, de elkészitésük nem az átlagos felhasználó feladata, két okból is. Egyrészt ezek is gyakori, ismétlődő feladatot oldanak meg, igy többszöri ujrairásuk felesleges, nem gazdaságos. Másrészt megirásukhoz általában az átlagos felhasználó tudása nem elegendő.

A fent körvonalazott software eszközök együtt alkotják azt a környezetet, amely a programozó-alkalmazó számára sokkal inkább jelenti a gép arculatát, tulajdonságait, korlátait, mint a gép hardware elemei. Igy a számitógép ismerete elsősorban a software-rendszer ismeretét jelenti.

3. ISMERKEDÉS A CP/M-MEL

Mint már a korábbiak során emlitettük, az operációs rendszer egy olyan program, amely egyebek között elősegiti a programok és adatok tárolását és visszakeresését a

háttértárolón. Mivel az operációs rendszer is egy lemezről betöltendő program, a CP/M betöltése a 22-es csapdájának látszik, hiszen azt egyszerüen csak a CP/M segitségével tudnánk megoldani. A megoldást az un. bootstrap loader /kezdeti betöltő/ program használata nyujtja. A személyi számitógépet ellátják gyártásakor egy ROM-ba beégetett /megváltoztathatatlan/ programmal, amelynek csupán annyi a feladata, hogy a lemez első sávjának első szektorát beolvassa a memória legelejére, amikor a gépet bekapcsolják, és ez a kis /egyszektoros/ program lesz az, amely már képes betölteni a lemezről magát az operációs rendszert. Ezt az eljárást nevezzük hideginditásnak /cold-boot/. Ez általában megismételhető a gép bekapcsolt állapotában is /általában a reset billentyü megnyomásával/. A későbbiekben majd megismerjük a CP/M egy másik inditását is, az un. melegindítást /warm-boot/; ilyenkor az operációs rendszer nagy része már a gépben van, csak a felhasználóval kapcsolatot tartó modult kell a memóriában felujitani, valamint ilyenkor a rendszer megvizsgálja és megjegyzi magának a bekapcsolt lemezegységekben lévő lemezek jellemzőit. Ezért, ha lemezt cserélünk /és a lemezre a későbbiekben irni is akarunk/, feltétlenül szükséges melegindítást végrehajtani. Ezt a legtöbb géptipuson a CTRL /control/ és a C billentyük egyidejü megnyomásával érhetjük el /jelölése a továbbiakban: CTRL-C/.

Mivel a gépek bekapcsolása és a CP/M rendszer betöltése gépenként különböző müveleteket igényel, legegyszerübb itt különválasztani az egyes tipusokat.

3.1. Commodore 64

Ez a gép rögtön egy speciális eset. Mivel a CP/M az Intel 8Ø8Ø és a Z8Ø CPU-val rendelkező gépek operációs rendszere és a C 64 651Ø-es tipusu mikroprocesszoron alapul, a gépen a CP/M közvetlenül nem használható. A problémát a következőképpen oldották meg:
a C 64-be /a gép hátulján lévő csatlakozóba/ behelyezhető egy un. Z8Ø-cartridge, amely egy Z8Ø CPU-t tartalmaz és behelyezés után szervesen kapcsolódik a gép rendszerébe. A CP/M alatti müködésnél a 651Ø-es CPU csak mint I/O processzor funkcionál, megfelelő interface-t nyujtva a Z8Ø felé.

A CP/M müködéséhez szükséges minimális konfiguráció:

- C 64 alapgép,
- Z8Ø cartridge,
- VC 1541 tipusu lemezegység /single vagy dual/,
- TV készülék vagy video monitor.

A fentieken kivül a rendszer természetesen tartalmazhat még tetszőleges Commodore perifériát. A bekapcsolás és rendszertöltés menete:

1. Összeállitjuk a berendezést. Valamennyi készülék kikapcsolt állapotában végezzük ezt a feladatot! A tápegység kábelét a C 64 jobboldalán lévő tuchel aljzatba, a TV kábelt a TV antennabemenetébe és a gép hátulján lévő antennakimenetbe dugjuk. A bus kábelét /7 pólusu tuchel/ a gép hátoldalán középen lévő /baloldali/ aljzatba és a lemez-egységbe dugjuk /ha van printer, ugyanilyen kábel vezet

oda is/. A CP/M cartridge-et a gép hátoldalán jobboldalt lévő aljzatba helyezzük.

2. Bekapcsoljuk sorban valamennyi perifériát és csak legutoljára a központi egységet!

3. A lemezegységbe helyezzük a CP/M rendszerlemezt. A lemezről LOAD "*",8 begépelésével betöltjük a bootstrap loadert, majd RUN-nal elinditjuk .A program futása /a tulajdonképpeni rendszertöltés/ alatt a képernyőn egy sor csillag látszik.

4. A töltés befejeztével a rendszer cimmel, majd A> prompttal jelentkezik be a képernyőn.

A töltés intelligensebb lebonyolitására itt az ad alkalmat, hogy a C 64 gép operációs rendszer nélkül is elég jól kihasználható szolgáltatásokkal rendelkezik. A gépen a CP/M rendelkezésére álló terület kb. 48 Kbyte, ebből a felhasználói programok 44 Kbyte-ot használhatnak. A kezelést megkönnyiti, hogy a gép 8 funkcionális billentyüje parancsok röviditésére szolgál:

f1 - DIR
f2 - DIR B:
f3 - STAT *.*
f4 - STAT B: *.*
f5 - COPY
f6 - CONFIG
f7 - DDT
f8 - DDT /+folytatás/

A billentyük megnyomása után a CR /carriage return - kocsi vissza/ billentyüt nem kell megnyomni, a parancs

azonnal végrehajtódik. Kivétel az f8, itt a funkcióbillentyü csak a parancs nevének begépelését helyettesiti, a folytatást a billentyüzeten kell megadni.

3.2. MØ8X és Proper 8

Ezeknek a gépeknek az operációs rendszere, a PROPOS 8 tulajdonképpen teljesen megegyezik a CP/M-mel.
A CP/M müködéséhez szükséges konfiguráció:

- központi egység
- video monitor
- MOM 8"-os, vagy 5 1/4"-os duál lemezegység.

A fentieken kivül lehetőség van egyéb perifériák, pld. printer kezelésére is. A gép kapható grafikus opcióval is, ez nem befolyásolja a CP/M használhatóságát.
Az üzembehelyezés menete:

1. Összeállitjuk a berendezést valamennyi egység kikapcsolt állapotában.

2. Bekapcsoljuk a monitort, a lemezegységet, majd végül a központi egységet a hátoldalon elhelyezett kapcsolókkal.

3. A képernyőn a következő felirat jelenik meg: "helyezz rendszerlemezt az A egységbe és nyomd le bármelyik billentyüt!".

4. Hajtsuk végre az utasitást, mire a rendszer a szokásos A> prompt-tal jelentkezik a képernyőn.

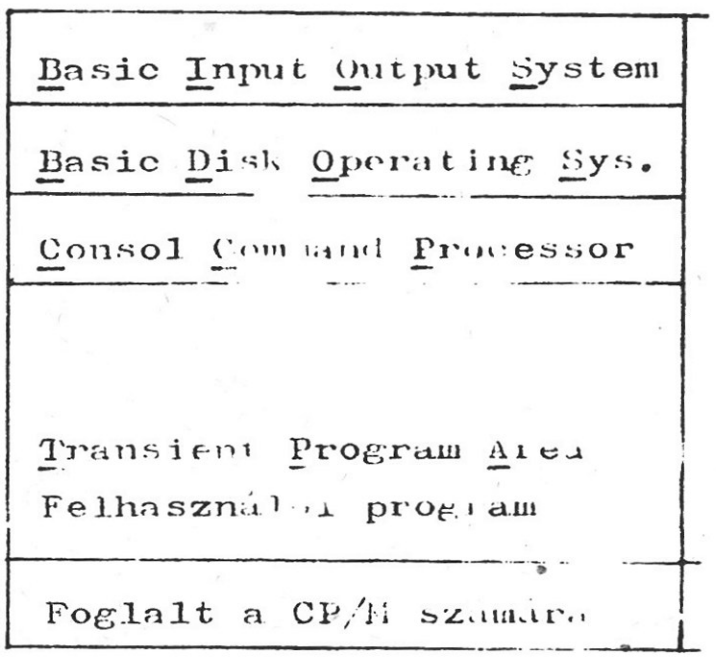
A PROPOS 8 operációs rendszer a CP/M valamennyi specifikációját és szolgáltatását kielégiti, bár megjegyzendő, hogy bizonyos tranziens parancsok csak külön árért vásárolhatók meg a forgalmazó cégtől.

II. RÉSZ:

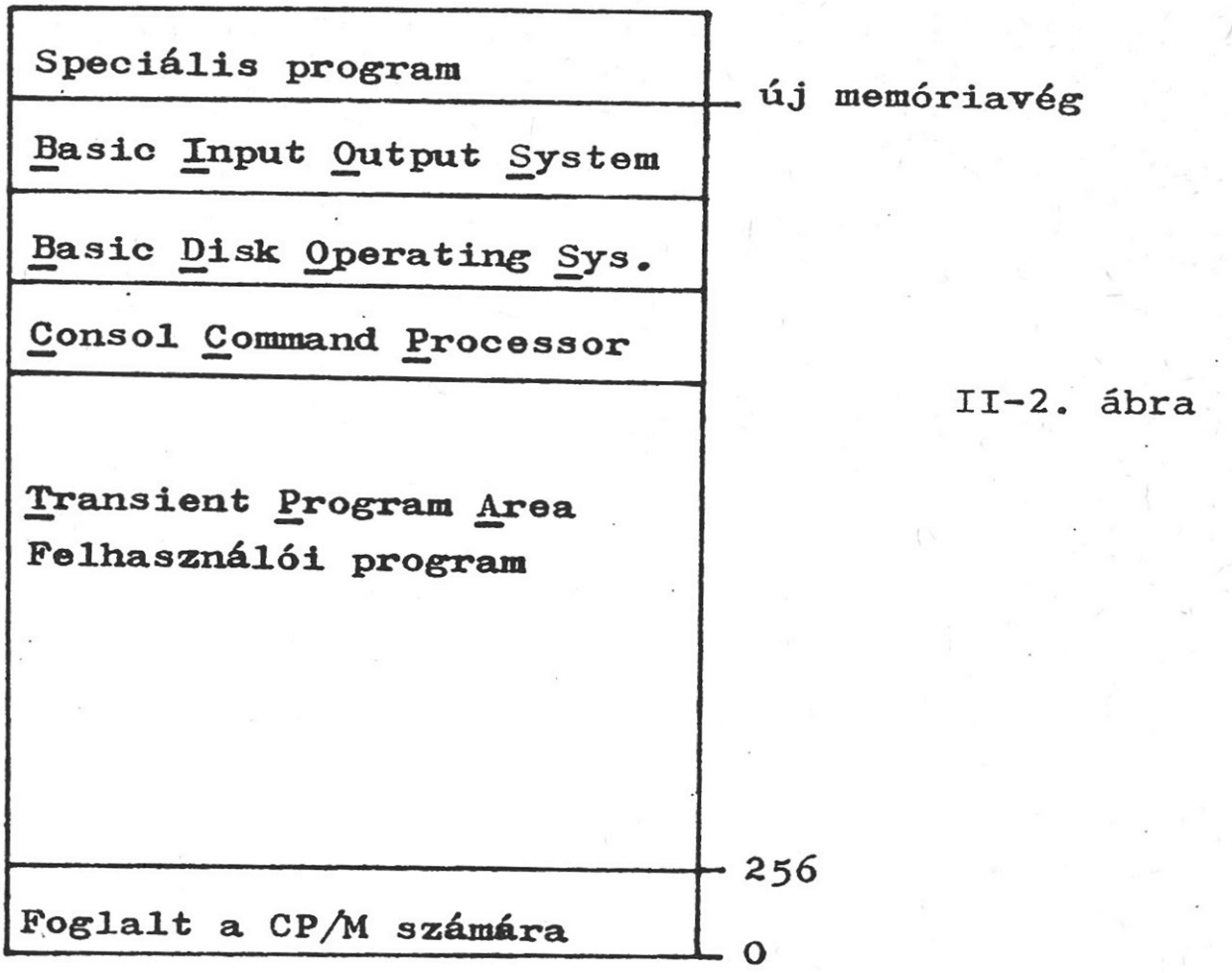
CP/M ALAPISMERETEK

1. MEMÓRIAFELOSZTÁS

Mint már az előző fejezetben emlitettük, a számitógépnek három fő része van, a CPU, a memória és az input/output eszközök. CP/M alatt a számitógépben a maximális cimezhető memória /legalábbis a közvetlenül cimezhető/ 65 536 byte, azaz 64K. /1 K = 1 kbyte; ez megfelel 1Ø24 byte-nak./ Valamennyi byte-nak saját cime van, amely decimálisan Ø-tól 65 536-ig, hexadecimálisan Ø-tól FFFF-ig terjedhet. A CP/M operációs rendszer a memóriát különböző rendeltetésü blokkokra osztja fel és bizonyos területeket saját céljaira foglal le. Ezek a területek mindig a rendelkezésre álló memória Ø-ás cimétől egy adott végcimig /top of memory/ kerülnek beosztásra, igy rugalmasan alkalmazkodnak az adott konfigurációban rendelkezésre álló memóriaméretehez. Ha a központi tár kisebb, mint 64 K, a végcim kisebb, a CP/M nem vesz tudomást az azon felüli cimekről, és a rendelkezésére álló kisebb területen helyezkedik el. Igy tulajdonképpen kétféle memóriafelhasználás képzelhető el a CP/M alatt, az egyiket a II-1., a másikat a II-2. ábra szemlélteti.



II-1. ábra



II-2. ábra

Láthatjuk, hogy az utóbbi esetében a CP/M tudomására hozott végcim kisebb, mint a ténylegesen rendelkezésre álló memória felsõ határa, igy az a fölötti rész nem CP/M alatt futó felhasználói programok futtatására használható fel.

Az ábrákon rögtön láthatjuk az operációs rendszer által lefoglalt, ill. szabadon hagyott területek elosztását is. A CP/M által lefoglalt elsõ 256 byte után a tranziens programok területe, a TPA /Transient Program Area/ következik és a BDOS által elfoglalt terület kezdetéig tart. Ez az a terület, ahol a felhasználói programok elhelyezkednek a CP/M felügyelete alatt álló memóriában. A CP/M a programokat erre a területre tölti be, a 256-os decimális cimtõl kezdõdõen byte-ról byte-ra, amig a teljes program a memóriába nem kerül, ill. tul nagy

program esetén addig, amig a TPA tetejét el nem érte. Bár az ábrán a BDOS és a TPA között még egy terület, a CCP-vel /Console Command Processor/ jelölt konzolparancs-feldolgozó helyezkedik el, ez a terület azonban a felhasználói programok számára rendelkezésre áll, azt felülirhatják. Ha a program befejezte müködését, a CP/M-hez való visszatéréskor az operációs rendszer a CCP-t a lemezről visszatölti /ezt a müveletet nevezzük meleginditásnak - warm boot/. A felhasználói programok azért használhatják a CCP területét, mert arra a program futása alatt nincs szükség, mivel az, mint később látni fogjuk, csak a felhasználó és az operációs rendszer közvetlen kapcsolatának eszköze. A TPA a CP/M memóriának az a része, amely rugalmas méretü, mig a többi terület hossza rögzitett. Ha a memóriaméret kisebb, mint 64 K, a CCP kezdete /s hasonlóképpen a fölötte elhelyezkedő többi területé/ kisebb cimre tevődik át, a TPA hossza lesz kisebb.

2. A CP/M RÉSZEI ÉS BELSŐ SZERKEZETE

Már az előzőekben megismerkedtünk a CP/M által felügyelt memória egyik részével, a tranziens programok számára fenntartott területtel, a TPA-val. A memória többi részét már nem a felhasználó által irott programok, hanem a CP/M részei foglalják el.

Az első 256 byte-on a rendszer legfontosabb közös változói, ugrótáblái helyezkednek el. Ennek a területnek a felhasználó szempontjából elsősorban mint a rendszerrel való kapcsolatteremtés eszközének van jelentősége. /Részletes leirása az V. részben található./

A TPA után a konzolparancs-feldolgozó, CCP helyezkedik el. Ennek kezdete a memória méretétől függ, elhelyezését /s a mögötte lévő CP/M részek elhelyezkedését is/ a SYSGEN és MOVCPM parancsok segitségével módosithatjuk, ezt majd a V/4. fejezetben részletesen tárgyaljuk. A CCP a CP/M-nek az a része, amely az operátorral való kapcsolattartást vezérli az un. parancs módban, tehát akkor, amikor felhasználói program nem fut és nem ragadta magához a vezérlést /mely esetben program módról beszélünk./ A CCP feladatairól a rezidens parancsok kapcsán még bővebben is lesz szó, most elégedjünk meg csupán annyival, hogy a CCP a CP/M és a felhasználó közötti közvetlen kapcsolattartásért felelős, programokat tölt be és indit el, file-okat listáz, hoz létre, töröl, valamint egyéb rendszerháztartási funkciókat lát el a rezidens parancsok segitségével.

A CP/M hátralévő két memóriaszelete az input/output tevékenységek lebonyolitására szolgál. Két részre bontása azért indokolt, mert a véletlen elérésü lemezkezelés alapvetően különbözik az egyéb I/O eszközök használatától, amelyek általában szekvenciálisak, vagy teljesen speciálisak.

A CCP után elhelyezkedő BDOS/Basic Disk Operating System/ - lemezes operációs rendszer végzi valamennyi lemezes file-tranzakció lebonyolitását, rekordok irását és olvasását, a lemezterület dinamikus lefoglalását és felszabaditását, valamint egyéb lemezkezelő tevékenységeket. A BDOS, eltérően a közvetlen I/O kezelést végző BIOS-tól teljesen gépfüggetlen, azaz független az adott géptipustól és konfigurációtól, vagyis, hogy milyen perifériák kapcsolódnak a számitógéphez. A BDOS tulajdonképpen a CP/M magja, amelyre az operációs rendszer minden szolgáltatása épül.

A már emlitett BIOS /Basic Input/Output System/ a CP/M operációs rendszer harmadik fő része, mégpedig az egyetlen olyan része, amely gépfüggő programokat tartalmaz. Igy a BIOS tartalmazza mindazokat az input/output programokat /driver-rutinokat/, amelyek a perifériák közvetlen kezelését végzik, igy a konzol-eszköz, a lemez-vezérlő interface és a printer kezelését is. Mivel a BIOS a CP/M többi része, ill. a felhasználói programok felé egy egységes kapcsolódási felületet képez, a BIOS/BDOS koncepció az, ami lehetővé teszi, hogy a CP/M szinte bármelyik mikroszámitógépen azonos módon fusson.

A CP/M szerkezetének a programozó számára talán legfontosabb tulajdonsága az, hogy a felhasználói program kapcsolatot tud teremteni a rendszer elemeivel. A felhasználói programból a rendszernek küldött parancsokat rendszer-hivásoknak nevezzük. A rendszer-hivások egy-egy "alacsony-szintü" funkciót valósitanak meg, pl. egy karakter beolvasását a billentyüzetről, vagy kiirását a képernyőre. A program egy gépikódu CALL utasitással aktivizálja a megfelelő rendszer-funkciót megvalósitó programot, amely a kivánt funkció teljesitése után visszaadja a vezérlést a hivó program soronkövetkező utasitásának. A felhasználói program megfelelő pontján, miután betöltöttük a megfelelő kommunikációs regiszterekbe a rendszer-hivás paramétereit és a megfelelő belépési pontot /funkció-sorszámot/ azonositó számot, mindig ugyanarra a cimre /hexadecimális ØØØ5/ való ugrással aktivizáljuk a CP/M-et. Igy pl. ha egy szöveget akarunk kiirni a képernyőre, vagy általánosabban szólva a konzol-perifériára, elhelyezzük a karaktersorozat cimét és hosszát, valamint a kivánt funkció sorszámát a megfelelő regiszterekbe, majd

kiadunk egy CALL utasitást a megfelelő cimre /ΦΦΦ5H/. Amikor visszakapjuk a vezérlést az operációs rendszertől, a karaktersorozat már kikerült a képernyőre. /A rendszerfunkciók részletes leirását az V/3. fejezet tartalmazza./ A fentiekből jól látszik a CP/M alatt futó programok gépfüggetlen jellege, hiszen a fenti folyamat nincs tekintettel arra, hogy valójában mi is az a bizonyos konzol-eszköz, milyenek hardware jellemzői, milyen vezérlést igényel, ezek a kérdések a BIOS gépfüggő részében dőlnek el.

A programok és a CP/M közötti információátadásra a CPU regiszterei szolgálnak, a C regiszter kiemelten a funkció-sorszám megjelölésére. Ha az átviendő információ mennyisége nagyobb, minthogy az a regiszterekben elférne, szabványositott paraméterblokkok segitségével történik az információcsere, s ilyenkor ezek cime kerül a regiszterekbe. Ennek legfontosabb esete a lemezes file-ok leirására használt file-leiró tábla, az FCB /File Control Block/. /Részletes leirását lásd az V. részben./

3. ÁLTALÁNOS TUDNIVALÓK A LEMEZRŐL

Mielőtt megismerkednénk a CP/M által kezelt lemezes file-okkal, elengedhetetlen, hogy néhány szót ejtsünk a lemezről általában.

Ma a lemezes tárolóknak két alapvető tipusa használatos: a hajlékony és a merev lemezes tárolók. Előbbiek olcsóbbak és könnyen kezelhetőek, utóbbiak viszont nagy

kapacitásuak és sokkal gyorsabbak. A hajlékony lemezes tárolókat diskette vagy floppy disc néven is szoktuk emlegetni. Két alapvető méretben készülnek, 8, ill. 5 1/4 inch-es kivitelben, utóbbit mini-diskette-nek szokás nevezni. A ma használatos mikroszámitógépek általában az olcsóbb floppy-lemezzel vannak felszerelve, s bár az utóbbi időben a mind olcsóbbá váló merev Winchester-tárolók egyre szélesebb körben kerülnek felhasználásra a mikroszámitógépeknél, a floppy általában ezeknél a konfigurációknál is használatban maradt.

A floppy lényegében egy kartontasakba zárt hajlékony müanyag korong, amelynek felületére vasoxid réteget vittek fel, ez a tulajdonképpeni mágnesezhető információhordozó. Használat közben a lemezegység motorjának tengelye a lemez közepén lévő lyukba hatol és azt nagy sebességgel forgatja a karton csomagoláson belül. A boritékon lévő hosszanti nyiláson keresztül tud a lemezegység iró/olvasó feje a lemezfelülettel érintkezni és igy információt rögziteni, ill. kiolvasni a lemezről. Ez a folyamat nagymértékben hasonlit ahhoz, amely a szalagegységeknél, vagy a szórakoztató elektronikában használatos magnetofonoknál játszódik le. A lemezen az információt koncentrikus körökön rögzitjük, amelyeket sávoknak,vagy angol szóval track-eknek hivunk. Minden sáv logikailag felosztható egyenlő hosszuságu részekre, un. szektorokra. A szektorokban tárolható byte-ok száma, a sávok száma és azokon belül az egyes szektoroké a tipusoktól függően változó. A szektorok mérete 128, 256 vagy 1Ø24 byte lehet, a CP/M általában 128 byte-os szektorokkal dolgozik. A leggyakrabban használt felosztások és irássürüségek 8 inch-es lemez esetén:

- 77 sáv lemezoldalanként, 26 db 128 byte-os szektor sávonként, /egyszeres sürüség/,

- ugyanennyi sáv, de 26-szor 256 byte sávonként, /dupla sürüség/,

- ugyancsak 77 sáv, de 8 db 1Ø24 byte-os szektorral sávonként /dupla sürüség/.

A lemezek mind egy-, mind kétoldalas kivitelben előfordulnak, utóbbi természetesen kétszer annyi információ tárolását teszi lehetővé. A fentieknek megfelelően egy 8" egyszeres sürüségü egyoldalas lemez 256 Kbyte tárolását teszi lehetővé, mig egy dupla sürüségü, 256 byte-os szektorok esetén 512 Kbyte, 1Ø24-es szektorok esetén 633 Kbyte-ét. A lemezek boritójának oldalán egy kis bevágás található, ez az irásvédelemre szolgál. A 8"-es lemeznél, ha a nyilást nem fedi aluminium fólia, nem lehet a lemezre irni. Az 5 1/4-es mini-floppy esetében viszont éppen ellenkezőleg, a fennlévő aluminium fólia védi a lemezt a véletlen felülirástól. Igy pl. a gyári programokat, rendszereket tartalmazó lemezek irásvédetten kerülnek forgalomba.

A lemezek kezelése némi gondosságot kiván. Soha ne érintsük kézzel a nyiláson keresztül a szabadon maradt területet. Óvjuk a portól, a mágneses mezőktől, amelyeket pl. a nagyobb teljesitményü motorok keltenek, mivel azok tönkretehetik a lemezt. Mindig ügyeljünk arra, hogy a lemezt helyesen tegyük be a lemezegységbe, kerülve az erőltetést. Az első ismerkedéshez a lemezzel ne a nehezen és drágán pótolható gyári rendszerlemezt, hanem annak

egy másolatát tanácsos használni. Ha bekapcsolás vagy kikapcsolás alatt a lemez a lemezegységben van, az ilyenkor fellépő nagy áramok átmenetileg megzavarhatják a vezérlő elektronikáját és hibás jeleket irhat a lemezre, felülirva az azon tárolt esetleg igen fontos információt. Ha a lemezeket mindig csak a bekapcsolást követően helyezzük be az egységbe és a kikapcsolás előtt kivesszük onnan, az áramkimaradásokat leszámitva nem fogunk találkozni ilyen problémával.

Mielőtt a lemezt igénybe vennénk valamilyen program vagy adat tárolására, formattálnunk kell azt, hacsak nincs már a gyártó cég által előre formattálva. A formattálás kettős feladatot lát el. Egyrészt a lemez valamennyi szektorát üresre állitja, egyuttal megvizsgálva használhatóságát is, másrészt beállitja az adott irássürüségnek és lemezfelosztásnak megfelelő formátumot, amely természetesen különböző irássürüségenként és a szektorhossztól függően. Ha ugy döntünk, hogy egy adott lemezt más irássürüséggel kivánunk használni, azt feltétlenül ujra kell formattálnunk. Az eredeti CP/M operációs rendszerhez a Digital Research nem szolgáltat ehhez standard rendszerprogramot; a különféle géptipusokhoz CP/M-et forgalmazó cégek a rendszerrel együtt egy ilyen funkciót megvalósitó tranziens programot is szállitanak, általában FORMAT-néven.

A formattálás CP/M alatt a FORMAT parancs begépelésével történik. Ezután a rendszer néhány kérdést tesz fel a formattálás paramétereire vonatkozóan, egyszeres vagy dupla sürüséget, 128, 256 vagy 1Ø24 byte-os szektorokat kivánunk-e használni, stb. A müvelet befejeztével a tranziens parancs visszaadja a vezérlést a CCP-nek, a

lemez használatra kész. A lemezen előzőleg tárolt információk természetesen ilyenkor elvesznek.

A CP/M felügyelete alatt dolgozó gépeknél kétféle lemezről beszélhetünk. Az egyiken csak file-ok vannak, mig a másikon a program, adat és szövegfile-okon kivül a CP/M egy másolata is található, utóbbiak a rendszerlemezek. Általában az A egység rendszerlemezt tartalmaz, mig a többi /B, C, stb./ egységekben akár rendszer-, akár egyéb lemez lehet. A rendszer-lemezek létrehozása a FORMAT és a SYSGEN rendszer-programok egymásutáni futtatásával lehetséges, mint majd a későbbiekből kiderül.

4. KOMMUNIKÁCIÓ A RENDSZERREL

Mint már az I/3. fejezetben emlitettük, a rendszer betöltése után az A> prompttal jelentkezik be a konzolperiférián, azaz általában a képernyőn, amellyel jelzi, hogy kész a parancsok fogadására. Ez az üzenet egyuttal azt is közli, hogy A az un. kurrens lemezegység, tehát amire a parancsok /hacsak nem jelölünk meg mást/ vonatkoznak. Prompt-nak nevezzük a rendszer által a képernyőre kiirt jelet, amely parancs begépelésére szólit fel.
Ha azt akarjuk, hogy a kurrens lemezegység a továbbiakban a B jelü legyen, egyszerüen irjuk:

A>B:

amire a CP/M a B> prompttal válaszol, jelezve, hogy

a B: parancs hatására átkapcsolt a B lemezre. Természetesen vigyáznunk kell arra, hogy csak olyan egységre hivatkozzunk, amely valóban van a rendszerünkben, mert pl. egy kétlemezes rendszerben a D-re való hivatkozás olyan állapotba hozza a rendszert, hogy onnan már csak hidegindítással lehet előcsalogatni.

A felhasználó a parancsok begépelésében segitséget kap a rendszertől a vezérlőkarakterek formájában. Ezek egy részének csak az irógépes konzolokon van jelentősége, mivel a képernyős berendezéseknél a képernyőszerkesztő funkciókkal kényelmesebben és áttekinthetőbben oldhatók meg ugyanazok a feladatok, amelyekre ezeket a vezérlőkaraktereket definiálták. A vezérlőkarakterek többi része viszont a képernyős konzolokon is fontos. A vezérlőkarakterek használata a CTRL /Control/ és a megfelelő betü egyidejüleg történő lenyomásával történik. Az editálási feladatokat ellátó vezérlőkarakterek a következők:

- CTRL-U vagy CTRL-X, törli az éppen begépelés alatt álló sort és a következő sor elején a kurzor villogtatásával jelzi, hogy uj parancs fogadására kész.

- CTRL-H, egy karakterpozicióval való visszalépést és az aktuális karakter törlését eredményezi.

- CTRL-R, kiirja az éppen begépelés alatt lévő sort, több javitó-vezérlőkarakter után jól használható.

- CTRL-E, a következő sor elejére áll és várja a sor folytatását; egy sornál hosszabb parancsok begépelése esetén a sor végén a RETURN helyett használatos, mivel az terminálná a parancsot.

A többi vezérlőkarakter fontos vezérlő funkciót lát el. Speciális közülük a RETURN vagy CR billentyü /amely általában CTRL-M-mel vagy CTRL-J-vel is helyettesithető/, ez a parancsok terminálására szolgál, hatására a rendszer végrehajtja a parancsot, majd a következő sor elején prompttal jelentkezik ki.

Egyéb vezérlőkarakterek:

- CTRL-C, az operációs rendszer meleg-inditását idézi elő, amit vagy az éppen futó folyamat megszakitására használunk, vagy egy uj lemez behelyezésekor, hogy azt a rendszer tudomására hozzuk.

- CTRL-P, megnyomásával a konzolon váltott üzeneteket a sornyomtatón listázza, ujabb CTRL-P a listázást kikapcsolja.

- CTRL-S, felfüggeszti a konzolra történő kiiratást, a program fut tovább; ujbóli megnyomására a kiirás folytatódik.

Mint már tudjuk, amikor a rendszer prompttal jelentkezik ki a képernyőn, ami egy betüből /kurrens lemezegység/ és egy nagyobb jelből / > / áll, parancsokat gépelhetünk be. A parancsok valamennyien - a rendszer-prompttól eltekintve - két részből állnak, utasitásból és argumentumból vagy argumentumokból. Pl. a következő parancs:

A>ERA PELDA.DAT

esetén az "ERA" az utasitás, mig a PELDA.DAT az argumentum /a parancs egyébként, mint később látni fogjuk, a

PELDA.DAT file törlését jelenti/. A CP/M CCP programja feltétlenül elvár legalább egy szóközt a parancsban az utasitás és az argumentumok között, de irhatunk több szóközt is. Hasonlóképpen használhatunk szóközt az "=" és az "A:" jelek után is, de általában másutt nem. Bizonyos CP/M implementációk megengednek szóközöket akárhol a parancsban, de ez általában nem igaz, többnyire a fent felsorolt helyeken kivüli szóközök terminálják a parancs szövegét vagy szintaktikus hibát okoznak.

Mielőtt gyorsan végigszaladnánk a parancsok egyes tipusaink, meg kell ismerkednünk a CP/M alapvető fogalmával, a lemezes file-ok rendszerével.

A CP/M legszebben kidolgozott része s népszerüségének elsődleges forrása file-kezelő rendszere. A file háttértáron tárolt adatok, forrásszövegsorok, vagy programutasitások halmaza. A file-okra legfeljebb nyolckarakteres file-névvel és háromkarakteres file-tipussal hivatkozhatunk, utóbbit kiterjesztésnek /extension/ is szokás nevezni. Valahányszor a file-ra hivatkozunk, kötelező megadni a file nevét és általában tipusát is. A tipusmegadási kötelezettség alól kivétel például a tranziens parancsok aktivizálása, ahol a kötelezően COM tipus elhagyandó. A CP/M , ill. bizonyos rendszerprogramok egyes file-oknál kötelezően előir-ják a file-tipust, más esetekben használata opcionális, értéke tetszőleges. A file neve tetszőleges ASCII karaktereket tartalmazhat a következők kivételével:

? * . , = :

A CP/M rendelkezik olyan, nagyon praktikus tulajdonsággal is, amely lehetővé teszi, hogy egyidejüleg a file-ok

egész csoportjára hivatkozzunk. Igy beszélhetünk egyedi és csoportos file-hivatkozásról. Az előbbi esetben a file-hivatkozás csak egyetlen, pontosan megnevezett file-t azonosit, mig az utóbbi jelenthet több file-t egyidejüleg. A CP/M ezt az un. joker karakterek segitségével teszi, amelyek ugyanugy, mint a joker a kártyajátékokban, egy sereg más kártyát - karaktert vagy karaktersorozatot helyettesithetnek. Két ilyen joker karakter van: a "*" és a "?" karakter helyén tetszőleges karakter állhat a file-névben. Igy pl. a következő file-okra a BEVETEL?.DAT file-névvel egyszerre hivatkozhatunk:

BEVETEL1.DAT, BEVETEL2.DAT, BEVETELE.DAT.

Ennek értelme, ha a file-okra vonatkozóan valamilyen CP/M parancsot akarunk megadni, elegendő egyszer begépelni a parancsot a fenti csoportos file-hivatkozással, amely azt jelenti, hogy a kivánt müveletet valamennyi file-lal végre kell hajtani.

Hasonló célt szolgál a "*" is, de ez nem egyetlen karaktert, hanem többet is helyettesit. Igy pl a következő parancs valamennyi BAS tipusu /BASIC/ file-t kilistázza a képernyőn, amely az A lemezen található:

A> DIR *.BAS

Bizonyos file-tipusok speciális jelentéssel birnak a CP/M, a rendszerprogramok, ill. a nyelvi rendszerek számára. Sok későbbi problémától menthetjük meg magunkat, ha ezeket a tipusneveket nem használjuk egyéb célra. A legfontosabb, szabványosnak mondható tipusok a következők:

.ASM - assembler forrásprogram

.BAK - ED által generált biztonsági másolat /backup/

.BAS - BASIC forrásprogram

.CRF - áthelyezhető keresztreferencia

.COB - COBOL forrásprogram

.COM - végrehajtható program /tranziens parancs

.DAT - ASCII adatfile /FORTRAN alapértelmezés/

.FOR - FORTRAN forrásprogram

.HEX - hexadecimális kódra forditott /assemblált assembler-program

.INT - BASIC tárgykód

.MAC - áthelyezhető assembler kód

.LST - BASIC lista

.OVR - COBOL fordító overlay

.PRN - lista /COBOL, FORTRAN ASM, stb./

.REL - áthelyezhető tárgykód /FORTRAN, ASM, stb./

.SUB - SUBMIT parancs-file

.SYM - ASM szimbólumtábla

.XRF - abszolut keresztreferencia

.$$$ - ED vagy PIP átmeneti file.

Egy CP/M file nulla byte adatot is tartalmazhat, vagy másik végletként kiterjedhet egy egész lemeznyi területre. Az operációs rendszer nagy előnye, hogy ha egy file számára területre van szükség, lefoglalja azt, de ha a file

kisebb lesz, a használaton kivüli területet felszabaditja, elérhetővé teszi más file-ok számára. Ezt az eljárást dinamikus file-allokációnak nevezzük, ami lehetővé teszi a lemez optimális kihasználását.

A lemezen lévő file-ok egy szintén a lemezen lévő directory-ban vannak nyilvántartva. Egy file-hoz több, legfeljebb 255 directory bejegyzés / entry/ és egy bejegyzéshez több, legfeljebb 16 file-darab, un. file-terület tartozhat./Egy file-terület általában Kbyte-okban mérhető. Erre a szervezésre azért van szükség, hogy a file-ok által lefoglalt lemezterület mérete dinamikusan változtatható legyen. Igy egy file nem egy összefüggő területet foglal el rendszerint, a file-területek azonban már összefüggőek./ A directory bejegyzései a file létrehozásakor keletkeznek, mig törlésük a file törlésekor játszódik le. Utóbbi müvelet természetesen a file-hoz tartozó szektorok felszabaditását is elvégzi.

Korábban igértük, hogy a CCP /konzolparancs-feldolgozó program/ müködését még részletesebben is tárgyaljuk. Most, megismerkedve a szükséges CP/M alapfogalmakkal, már megvan erre a lehetőségünk.

A CCP a CP/M-nek az a része, amellyel a felhasználó a leggyakrabban kapcsolatba kerül. A CP/M kétféle parancsot ismer, rezidens és tranziens parancsokat. A rezidens parancsokat végrehajtó programok a CCP részei és állandóan a memóriában találhatók /legalábbis parancs módban/. A tranziens parancsokat megvalósitó programok a lemezen tárolódnak és nyilván csak akkor hivhatók, ha az őket tartalmazó lemez éppen valamelyik lemezegységben van. A tranziens programokat gyakran rendszer/segéd/programoknak is nevezzük.

A CCP a már megismert "A> " prompttal jelzi, hogy kész a rezidens parancsok végrehajtására. A CP/M öt rezidens parancsot ismer, ezek minden változatban megtalálhatók, mig a tranziens parancsok készlete rugalmas, bővithető vagy szükithető az adott igényeknek megfelelően.

A DIR parancs talán a leggyakrabban használt valamennyi rezidens parancs közül. Segitségével a felhasználó megállapithatja, milyen file-ok vannak valamely lemezén. Kilistázható vele valamennyi file neve, csoportos file-hivatkozást használva file-ok egy csoportjának neve, vagy - szélső esetben - egyetlen file neve. Utóbbi esetet arra használjuk, hogy megállapitsuk, rajta van-e egyáltalán a kérdéses file a lemezen. A DIR segitségével a kurrens lemezegység megváltoztatása nélkül más lemezek tartalmát is megvizsgálhatjuk.

Mig a DIR parancs csak a file nevét és tipusát irja ki, addig a TYPE a teljes file-tartalmat kilistázza a képernyőn. Megszoritás, hogy a file csak ASCII-file, tehát szöveges file lehet, és csak egyedi file-hivatkozással aktivizálható. Van azért lehetőség több file egyetlen paranccsal való kilistázására is, mivel vesszővel elválasztva több egyedi file-hivatkozást is felsorolhatunk.

Az ERA vagy ERASE parancs egy vagy több file-t képes törölni a lemezről. Az utóbbi esetben csoportos file-hivatkozással érjük el, hogy a parancs egyszerre több file-ra vonatkozzék. Mivel a "*.*" jogos hivatkozás, segitségével a lemez valamennyi file-ját kitörölhetjük. Az ilyen parancsot mindenesetre tanácsos egy DIR paranccsal megelőzni, nehogy fontos file-okat semmisitsünk meg.

A REN funkció segitségével megváltoztathatjuk egy file nevét és/vagy tipusát, a file valamennyi egyéb tulajdonságának /tartalmának, lemezen elfoglalt helyének/ változatlanul hagyásával.

A SAVE parancs elsősorban ASM /assembler nyelvü/ programok lemezre tárolásában nyujt segitséget. Müködését és funkcióját majd az assembly programozäs tárgyalásánál érthetjük meg.

A tranziens programok olyan funkciókat valósitanak meg, amelyek a rezidens funkciókkal ellentétben aránylag ritkán szükségesek, ill. olyan sok memóriára van szükségük, hogy rezidens tárolásuk nem oldható meg. A tranziens parancsokat programok valósitják meg, amelyek .COM tipusu file-okban a lemezen találhatók, és hivásukkor a CCP tölti be azokat a TPA-ba. Az alapvető tranziens parancsok a következők:

STAT - lemez- és file-információk

PIP - file-ok másolása és áthelyezése

SYSGEN-
és
MOVCPM- rendszergenerálás és módositás

SUBMIT- batch-feldolgozás

ED - editor

ASM - assembler forditóprogram

DDT - dinamikus program-"belövő" program

LOAD - program betöltése

DUMP - memória kiiratás.

A fenti parancsokon kivül még egy sereg program használható tranziens parancsként, igy a különböző nyelvi fordítók, általános funkciókat ellátó rendszerprogramok, de ezek választéka nem minden alaprendszerben áll rendelkezésre, többnyire külön kell megvásárolni. A CP/M rendszerprogramok terjesztésében óriási a szerepe a CP/M felhasználók körének /CP/M User's Group, vagy CMUG, 164 West 83rd Street, New York, NY 10024/. A szervezet tagjai lényegében az adathordozó áráért jutnak hozzá a legujabb CP/M programtermékekhez.

5. ALAPVETŐ REZIDENS PARANCSOK

Most ismerkedjünk meg egy kicsit részletesebben, példákon keresztül az előzőekben felsorolt rezidens parancsok használatával.

5.1. DIR /Directory/

A DIR parancs a lemezeken levő file-ok listájának megjelenitésére szolgál. Az A lemezegységben lévő lemez file-jainak listázása:

A>DIR↵

A B: lemezegységen levő file-ok listázásához a

A>DIR B:↵

utasitásra van szükség.

Ha azt szeretnénk, hogy a B legyen a kurrens lemezegység, a

```
A>B:↵
B>DIR↵
```

módszerre van szükség.

A parancs alkalmas lemezen lévő file keresésére is.

```
A>DIR B:BIMBÓ.TXT↵
```

A példán a B lemezen lévő BIMBÓ.TXT file létezéséről szeretnénk meggyőződni. Kilistáztathatjuk az összes .TXT tipusu file-t is:

```
A>DIR B:*.TXT ↵
```

A DIR utasitást gyakran használják ellenőrzésre is. Egy file törlése vagy létrehozása után meg kell győződni a directory helyességéről. A CP/M 2.2 változatában a directory listázása négy oszlopban történik.

A Commodore 64 gépen a DIR parancsra a következő listát kapjuk:

```
A>DIR↵
A: MOVCPM COM        TYPE COM
   CUBMIT COM        XSUB COM
   ED     COM        ASM  COM
   DDT    COM        LOAD COM
   SYSGEN COM        COPY COM
   CONFIG COM        STAT COM
   DUMP   COM
```

5.2. TYPE

A TYPE parancs egy ASCII-file képernyőn való megjelenitésére szolgál. Az utasitás általános formája:

TYPE d:név.tipus

ahol d a lemezegység azonositója, ez nem kötelező paraméter. Ha a file-t ki is szeretnénk nyomtatni, a parancs előtt CTRL-P-t kell billentyüznünk. A nyomtató használatával a képernyő lelassul.

A TYPE a számitógép memóriájában tárolt formában nyomtatja ki a file-t. A parancs bármilyen létező szöveg--file vizsgálatára alkalmas. Jó eszköz ujonnan létrehozott file-ok ellenőrzésére.

5.3. REN /Rename/

A REN parancs lehetővé teszi egy file nevének a megváltoztatását. Általános alakja:

REN d: uj név.uj tipus=d:régi név.régi tipus

azol a d lemezegység-azonositó opcionális. Nézzünk néhány példát. Változtassuk meg az A lemezen lévő TRANS.TXT nevü file nevét:

A>REN MAT.TXT=TRANS.TXT ↵

A TRANS.TXT-t elneveztük MAT.TXT-nek, a TRANS név megszünik, maga a file tartalma érintetlen marad.

A következő példában az A a kurrens lemezegység, a változtatást azonban a B lemezegységben levő lemezen szeretnénk elvégezni.

A > REN B:AGFA.TXT=AGFA.BAK↵

és a

A > REN B:AGFA.TXT=B:AGFA.BAK↵

utasitások jelentése megegyezik. A lemezegység azonositójának a meghatározása az egyenlőségjel bármelyik oldalán lehetséges. Szokásos módszer, hogy a file nevében utalunk a létrehozás vagy az utolsó hozzáférés dátumára:

CIMAUG8.

5.4. ERA /Erase/

Az Erase parancsot file vagy file-ok törlésére használhatjuk. Általános alakja:

ERA d: file-név.tipus

ahol a d lemezegység-azonositó opcionális. Az első példában a RAPID.TXT file-t töröljük.

A> ERA RAPID.TXT↵

Az utasitás egyszerre több file törlésére is alkalmas. Töröljük az összes .BAK-ra végződő file-t.

A ERA *.BAK↵

A file neve helyére a * jelet kell helyettesiteni /ld. a II/4. fejezetben a csoportos file-hivatkozások fajtáit/. A * jel tetszőleges számu karaktert jelenthet, beleértve a Ø-t is. Ebben az esetben a *.BAK megfelelhet a MINTA.BAK , MÁSIK.BAK , 5.BAK , .BAK , stb. file-oknak.

A gyakorlatban sokszor van szükség file-ok módositására, ilyenkor a feleslegessé vált file-okat feltétlenül törölni kell.

5.5. SAVE

A SAVE parancs a TPA meghatározott részét egy általunk definiált file-névvel ellátva a lemezre irja.
Általános formája:

SAVE pd:file-név.tipus

ahol d lemezegység azonositó, nem kötelező paraméter. A p/decimális szám/ memória lapszámot jelent, melyet a DDT R parancsának a segitségével számithatunk ki. Az utasitás a memória 1ØØH /hexadecimális/ cimétől kezdve /TPA kezdete/ p számu lapot /1 lap=256 byte/ vesz figyelembe. Nézzünk néhány példát a p értékének kiszámitására. Az R parancs hatására a NEXT oszlop alatt leolvashatjuk a TPA-ban levő program után következő első üres hely memóriacimét. Tehát ha a megjelenitett értékből 1-et levonunk hexadecimális aritmetikában, megkapjuk az utolsó TPA cimet, melyet átkonvertálhatunk decimálisra.

- Tegyük fel, hogy a NEXT cim két darab Ø-ra végződik pl. 13ØØH. Vonjunk ki 1H-t, igy 12FFH-t kapunk, majd hagyjuk el az utolsó két jegyet, FF-et, és váltsuk át a 12H-t decimálisra. p=18-at kapunk. Ez a 1ØØH és 12FF közé eső memória terület lapszáma.

- A nem két Ø-ra végződő NEXT cimekből nem kell kivonni 1H-t, hanem egyszerüen az első két jegyet konvertáljuk át decimálisra. Például a 15Ø6H-ból 21 lap lesz. Nagyon könnyen megoldható a feladat , ha a DDT-ben a NEXT alatt 3FFH-t kapunk. A lapra való konvertáláshoz hagyjuk el az FF-t. A 3H decimálisra átkonvertálva is 3. Ezért a SAVE utasitásban p=3-at lehet használni.

- Nézzünk egy sokkal összetettebb feladatot. A DDT R parancsának hatására a NEXT alatt megjelenő érték 1DØØH.

1DØØH-1H=1CFFH

A lapra való konvertáláshoz a két magasabb helyiértéket használjuk.

1CH = 28 decimálisan

Igy p=28 megfelel a 1ØØH /TPA kezdete/ és 1CFFH cimü memória közé eső lapszámnak.

Nézzünk néhány példát a SAVE parancsra.

A>SAVE 6 ANITA.COM↵

Tárolja a memóriát 1ØØH-tól 6FFH-ig és az A lemezegységen az ANITA.COM file-ba teszi. A 6FFH után a következő cim a 7ØØH.

A > SAVE 6ØB:ANDRIS.COM ↵

A B lemezegység ANDRIS.COM file-jába tárolja a memóriát a 1ØØH-tól a 3CFFH-ig. A következő cim a 3DØØH.

6. ALAPVETŐ TRANZIENS PARANCSOK

A CP/M tiz standard tranziens parancsa: SYSGEN, STAT, SUBMIT, PIP, ED, ASM, DDT, LOAD, DUMP, MOVCPM.
A lemezen ezek .COM file-ként szerepelnek. A parancsok közül jelen fejezetben csak az első hármat tárgyaljuk mélyebben.
A többi parancsnak itt csak rövid bemutatására vállalkozunk. Részletes ismertetésük később, külön fejezetekbe sorolva történik majd meg.

6.1. SYSGEN

A SYSGEN parancs rendszerlemezek másolására szolgál. Müködése során a rendszert behozza a memóriába /TPA/, majd átirja a meghatározott lemezegységben elhelyezett lemezre.
Formája

A > SYSGEN ↵

Nézzünk egy példát az utasitás használatára:

A > SYSGEN ↵

SOURCE DRIVE NAME: A ↵

```
DESTINATION DRIVE NAME:  B↵

DESTINATION ON B,THEN TYPE RETURN  ↵

DESTINATION DRIVE NAME:  ↵

A>
```

A SYSGEN segitségével a rendszer átmásoljuk az A lemezről a B-re. A parancs beirása után a rendszer a képernyőn kéri a forrás-lemez azonositóját, a példánkban ez az A egység. Megadhatjuk bármely olyan lemezegység azonositóját, amely rendszerlemezt tartalmaz. A SYSGEN ezután megkérdezi, hogy melyik lemezegységre másolja át a rendszert /DESTINATION DRIVE NAME:/. Bármilyen lemezegység azonositót megadhatunk, függetlenül attól, hogy abban a rendszerlemez van-e vagy sem. Ezután egy ellenőrző kérdés következik, mely segit az esetleges tévedés kiküszöbölésében. Kiiródik a kért cél-lemezegység azonositója; ha megfelel, a rendszer egy CR hatására végtehajtja a másolást. A SYSGEN ezután lehetővé teszi további másolatok készitését és ismételten megkérdezi a cél-lemezegység azonositóját mindaddig, mig csak erre a kérdésre CR-választ nem kap. Ekkor visszatér a CCP-be.

Célszerü a CP/M rendszerlemezről az első használatbavétel alkalmával azonnal másolatot készitení és biztonságos helyre tenni.

6.2. STAT

A STAT az a tranziens parancs /azaz COM tipusu file-ban levő rendszerprogram/, melynek segitségével a felhasználó információt szerezhet a CP/M file-ok

állapotáról, STAT-usáról; ennek segitségével állapitható meg például a file-ok mérete. Nagyon sokoldaluan használható program: lehetővé teszi lemezek, file-ok, input/output egységek lekérdezését és más rendszer--funkciókat. A STAT használatának előfeltétele, hogy a rendszer valamely lemezegységén létezzék a STAT.COM file. Ezt ugy ellenőrizhetjük, hogy kiadjuk az alábbi parancsot:

```
A>DIR STAT.COM

STAT.COM
```

válasz esetén a STAT rendelkezésünkre áll, a STAT.COM file jelen van a rendszerben, a kurrens lemezegységen. Ha a CP/M nem találja a STAT.COM file-t ezen a lemezen, az alább látható módon válaszol:

```
A>DIR STAT.COM

NOT  FOUND
```

Ez azt jelenti, hogy a STAT.COM file pillanatnyilag nem található az A lemezegységen. Próbálkozzunk hát a B, C, stb. egységekkel /ha több lemezegységünk van/ mindaddig, mig meg nem találjuk a STAT.COM file-t. Ha mindezek után még mindig nem találtuk meg a STAT.COM file-t, keressünk a lemezeink között egy olyat, amin rajta van a STAT.COM, és helyezzük azt be az egyik /célszerüen az A/ lemezegységbe. Ha végül sikerült megtalálnunk, készen állunk arra, hogy a STAT segitségével mintegy "felfedezzük" rendszerünket.

A STAT és a lemezes file-ok

A STAT közli bármely adott lemezen szereplő file-ok számát, méretét és tipusát, és a lemezen lévő szabad helyek méretét. /Bizonyos szempontból a STAT a DIR--parancs kifinomultabb verziójának tekinthető./

A STAT a többi külső parancshoz hasonlóan egy parancssort fogad, értelmez és hajt végre. A parancs-sort közvetlenül a parancs neve után billentyüzzük be, majd a teljes parancs-sor begépelése után a CR /kocsi-vissza/ billentyü megnyomásával zárjuk le.

A STAT öt különböző, lemezzel kapcsolatos kérdést /kérést/ fogad el a parancs-sorban. Ezeket, funkciójukkal együtt, az alábbiakban foglaltuk össze:

Az öt, lemezzel kapcsolatos kérés
és funkciók:

STAT — Ha egy üres /paraméterek nélküli/ parancssort adunk meg /azaz, a STAT billentyüzése után közvetlenül a CR billentyüt ütjük le/, a STAT válaszképpen közli velünk a kurrens--lemezen lévő szabad lemezkapacitást.

STAT x: — Az üres parancs-sor variációjának tekinthető; ilyenkor megadjuk, hogy melyik lemezegységre vagyunk kiváncsiak. A STAT ilyenkor az x lemezegységben elhelyezett lemezen lévő szabad hely méretét közli velünk.

STAT efh — Ha egyedi file-hivatkozást adunk meg a parancs-sorban a STAT után /efh/, a STAT közölni fogja velünk a file méretét mind byte-okban, mind rekordokban mérve.

STAT csfh — Az előző parancs egy változata, melynek segitségével egyidejüleg file-ok egy csoportjáról kérhető információ a képernyőre. Az csfh /csoportos filehivatkozás/ lehet például *.COM, melynek hatására megkapjuk az összes .COM-tipusu file listáját.

STAT x:=R/O — Ennek a parancs-sornak a megadásával a felhasználó közölheti az operációs rendszerrel, hogy az egész lemezt csak olvasásra /és irásra nem!/ lehet használni. Alkalmazásakor járjunk el körültekintően, és ne feledjük, hogy a parancs által szabott megkötést csak a rendszer ujrainditásával lehet feloldani /ami lehet "meleg"-, vagy "hideg"-inditás/.

Példák:

```
A > STAT A:↵

BYTES REMAINING ON A: 185 K
```

A STAT paranccsal itt megkérdezzük, hogy mennyi szabad helyünk van még az A: lemezegységben lévő lemezen. A CP/M válasza szerint a szabadon felhasználható lemezkapacitás 185 kilobyte. Második példánkban már részletesebb információkat szolgáltat az operációs rendszer;

közli a pillanatnyilag a B: lemezegységben elhelyezett lemezen lévő összes file főbb jellemzőit.

```
A>STAT B:*.*↵

RECS  BYTS  EXT  D:FILENAME.TYP
 75    36K   3   B:ASM.COM
 5Ø    25K   2   B:DDT.COM
 3Ø    14K   1   B:ED.COM
  5     1K   1   B:LOAD.COM
 13     8K   1   B:PIP.COM
 25    12K   1   B:STAT.COM

BYTES REMAINING ON B: 185K
```

RECS jelöli a file-okban helyet foglaló rekordok számát, BYTS a file-ok méretét adja meg kilobyte-okban, EXT /extent/ jelentése, hogy hány directory-entry tartozik a file-hoz, mig a D:FILENAME.TYP rendre a lemezegységet, a file nevét és tipusát jelöli.

A CP/M 2.Ø és 2.2 változata ettől a formátumtól kissé eltér. A Commodore 64-es gépen például az alábbiakban bemutatott képernyő-elrendezés szerint jelennek meg az információk:

```
A>STAT *.*↵

Recs    Bytes    Ext    Acc
 64      8k       1      R/W      A: ASM.COM
 15      2k       1      R/W      A: CONFIG.COM
 11      2k       1      R/W      A: COPY.COM
 38      5k       1      R/W      A: DDT.COM
```

```
3       1k      1       R/W     A: DUMP.COM
52      7k      1       R/W     A: ED.COM
14      2k      1       R/W     A: LOAD.COM
82      11k     1       R/W     A: MOVCPM.COM
58      8k      1       R/W     A: PIP.COM
41      6k      1       R/W     A: STAT.COM
10      2k      1       R/W     A: SUBMIT.COM
7       1k      1       R/W     A: SYSGEN.COM
6       1k      1       R/W     A: XSUB.COM

Bytes Remaining On A:78k
```

Az Acc /access=hozzáférés/ oszlopában levő R/W a file hozzáférési módját adja meg; jelen esetben irható és olvasható file-okról van szó /R-read, W-write/.

Ha a lemezen több file van, mint ahány sor a konzol--egységünkön, a file-ok listája szaladni fog felfelé a képernyőn. Ha ez a helyzet, CTRL-S lenyomásával merevithetjük ki átmenetileg a képet. Ha tovább akarjuk folytatni a listázást, ez ujabb CTRL-S adásával érhető el.

A STAT következő alkalmazása egy lemezegység irás ellen való védelmére ad lehetőséget

```
A>STAT B:=R/O↵
```

R/O jelentése: read only, azaz csak olvasás. Ha egy lemezegységet R/O-nak nyilvánitottunk, minden ezt követő próbálkozáskor, amikor irni akarunk arra a lemezre, vagy file-t törölni róla, hibaüzenetet kapunk, és

a próbálkozás eredményeképpen a CP/M meleg-inditást hajt végre saját magán, mint minden olyan esetben, amikor valamilyen operációs rendszer hiba lép fel.

A CP/M 2.2 és 2.Ø változatánál négyféle attributomot adhatunk meg:

R/O	read only	csak olvasható
R/W	read/write	irható/olvasható
SYS	system	rendszer-file, nem fog szerepelni a directory listájában
DIR	directory	érvényteleniti a korábban kiadott SYS attributum hatását.

I/O /input-output/ perifériák és a STAT: speciális alkalmazások

A STAT az előbbiekben leirt, lemezzel kapcsolatos funkciók megvalósitása mellett lehetővé teszi a felhasználó számára azt is, hogy megváltoztassa a rendszerben előre beállitott logikai-fizikai periféria-megfeleltetéseket. A periféria logikai neve azt tükrözi, hogy a felhasználó egy adott perifériát mire használ. A periféria fizikai nevére a CP/M konfigurációfüggő részében, a BIOS-ban van szükség, a ténylegesen használt perifériák vezérléséhez. A BIOS belső felépitését az V. részben tárgyaljuk majd.

A CP/M négy logikai egység-név használatát teszi lehetővé a felhasználó számára, melyeket tetszőleges számu fizikai input-output egységhez, perifériához rendelhet hozzá a felhasználó.

Logikai egységek: CONSOL /konzol/, LIST /listázó/, READER /olvasó/ és PUNCH /lyukasztó/. Az első kettő a fontosabb, jelölésük: CON: illetve LST:
A másik kettő jelölése RDR: illetve PUN:

Az aktuális fizikai input-output egységek, perifériák vezérlését speciális, un. driver, vagy vezérlő-programok végzik. Ezeket a programokat, vagy elterjedtebb nevükön driver-rutinokat a BIOS foglalja magában.

A fizikai input-output egységek különféle logikai egységekként kapcsolhatók az operációs rendszer közreműködésével a számitógéphez. Ha például a BIOS tartalmazza a képernyő-egységet, egy nagysebességü sornyomtatót és egy modem-et /modulátor-demodulátor/ vezérlő szubrutinokat, a CON: logikai egység választható a modem vagy a terminálképernyő közül, mig a logikai LST: egység a három közül bármelyik lehet.

Az alábbiakban felsoroljuk a standard fizikai neveket és jelentésüket:

TTY:	Teletype-egység /kissebességü konzol/
CRT:	Képernyő-egység /nagysebességü konzol/
UC1:	Felhasználó által definiált konzol
PTR:	Papirszalag-olvasó
UR1:	Felhasználó által definiált olvasó 1
UR2:	Felhasználó által definiált olvasó 2
PTP:	Papirszalag-lyukasztó
UP1:	Felhasználó által definiált lyukasztó 1
UP2:	Felhasználó által definiált lyukasztó 2
LPT:	Sornyomtató
UL1:	Felhasználó által definiált listázó egység

Ez igy ugyan egy kicsit soknak tünik ahhoz, hogy megjegyezzük, de erre nincs is feltétlenül szükség: a CP/M segitségünkre siet, ha megkérjük rá. A STAT kiirja a képernyőre az összes lehetséges logikai-fizikai egymáshoz-rendelést; mindössze annyit kell tennünk ehhez, hogy - mint azt az alábbi példa is mutatja - kiadjuk a STAT VAL: parancsot:

```
A>STAT VAL:↵

CON:=TTY:CRT:BAT:UC1:

RDR:=TTY:PTR:UR1:UR2:

PUN:=TTY:PTP:UP1:UP2:

LST:=TTY:CRT:LPT:UL1:
```

Mindebből megtudjuk, többek között, hogy a konzol-egység lehet TTY:, CRT:, BAT:, UC1: driver-rutinokhoz rendelt fizikai egységek valamelyike, de például már nem lehet a sornyomtató egység /LPT:/.

A felhasználó a jelenlegi hozzárendelések bármelyikét, vagy akár az összeset megváltoztathatja; ehhez mindössze annyit kell tennie, hogy bebillentyüz egy olyan parancs-sort, ahol az egyenlőségjel bal oldalán a négy logikai egység egyikének azonositója szerepel, mig az egyenlőségjel jobb oldalán valamelyik megengedett, standard névvel hivatkozott fizikai inpu-output egység nevét adja meg. Például a STAT CON:=CRT: parancs a KONZOL-egységhez a terminál-képernyőt és klaviaturát rendeli hozzá. Igy tehát ahhoz, hogy kiirassuk az aktuális input-output egymáshozrendeléseket, azok

közül egyet megváltoztassunk, majd ujra kiirassuk a - most már módositott - egymáshoz-rendeléseket, a következőképpen kell eljárnunk:

```
A>STAT DEV:↵

CON:=CRT:
RDR:=TTY:
PUN:=TTY:
LST:=TTY:

A>STAT LST:=LPT:↵

A>STAT DEV:↵

CON:=CRT:
RDR:=TTY:
PUN:=TTY:
LST:=LPT:
```

A CP/M 2.ϕ és 2.2 változatában a STAT VAL-ra adott válasz kissé eltér a korábban bemutatottaktól:

```
Temp R/O Disk :d:=R/O

Set indicator :filename.typ $R/O $R/W $SYS $DIR

Disk Status   :DSK:  d:DSK:

User Status   :USR:

Iobyte Assign :

CON:          =TTY:   CRT:    BAT:    UC1:
RDR:          =TTY:   PTR:    UR1:    UR2
PUN:          =TTY:   PTP:    UP1:    UP2:
LST:          =TTY:   CRT:    LPT:    UL1:
```

Az utolsó négy sor megegyezik az 1.4 változattal. A fölötte levő sorok a lehetséges STAT utasitásokat és azok változatait mutatják be. Az R/O-ról és az indikátorokról /file-attributumokról/ már korábban volt szó. A kurrens vagy egy másik lemezegység státuszának megjelenitésére a STAT DSK: vagy a STAT d:DSK használható:

```
A>STAT DSK:↵

A:       Drive Characteristics
3888:    128 Byte Record Capacity
 486:    Kilobyte Drive Capacity
 128:    32 Byte Directory Entries
 128:    Checked Directory Entries
 128:    Records/Extent
  16:    Records/Block
  52:    Sectors/track
   2:    REserved Tracks
```

Ezen a példán egy dupla sürüségü lemez jellemzőit láthatjuk. A teljes record kapacitás 3888 rekord. A rekord az az adatmennyiség, amelyet egy I/O-müveletnél a rendszer valójában mozgat. A lemez kapacitása 486 Kilobyte, de ez nem tartalmazza a lemez rendszert tartalmazó területeit. A 32 byte-os directory entry-k száma 128, ezek a file-okról a memóriában tárolt FCB /File Control Block/-beli információ nagyrészét tartalmazzák. Egy file-t esetleg több directory entry ir le, egy directory entry-hez a példabeli lemezen 128 rekord /1 rekord 128 byte/ tartozhat. Az egy directory entry-hoz tartozó "file-rész" is még több, nem összefüggően elhelyezkedő egységből állhat a lemezen, ezek az egységek az un.

file-területek /v.blokkok/; ezek már összefüggőek. A blokkméret egy adott lemezre jellemző konstans, példánkban 16 rekord, azaz 16*128 byte. A szektor a CP/M-nél 128 byte, esetünkben egy sávon /track-en/ 52 van belőle. A lemezen 2 fenntartott sáv van. A fenti fogalmakat- - directory entry, FCB, file-terület az átlagos felhasználónak nem kell pontosabban ismernie. Rendszerprogramozók számára részletes magyarázat az V. részben található.

6.3. SUBMIT

Időnként hasznos és szükséges a CP/M parancsok sorozatát ugy végrehajtani, mintha egy program utasitásai lennének. Pl. ha egy operátor gyakran használ egy utasitássorozatot /parancs-sorozatot/, célszerü az utasitássorozatnak nevet adni, s azt egy egyszeri utasitással végrehajtani ugy, mint egy programot. Ezt teszi lehetővé a CP/M a SUBMIT tranziens paranccsal. A SUBMIT feltételez egy .SUB végződéssel rendelkező file-t, amely tényleges utasitássorokat tartalmaz esetleg argumentumokkal, amelyek végrehajtáskor értékekkel helyettesitendők. A .SUB file ugy készül, mint egy szöveg-file, ED vagy bármely más szerkesztő-programot használva, és az utasitássorokat ugy sorolja fel, mint ahogy azokat a terminálon gépelnék, pl. MINTA.SUB tartalmazhatja a következő sorokat:

```
DIR $1:$2

PIP A: =$1:$2
```

A $1 és a $2 argumentumok. Ezek olyanok, mint egy

programban a változók - tényleges értékekkel lesznek helyettesitve, amikor azokat a végrehajtáskor /azaz amikor a SUBMIT utasitást használjuk/ megadjuk. Itt a $1 argumentumot a lemezegység betüjével helyettesitjük, a $2 argumentumot pedig egy teljes file-névvel /kiterjesztést is beleértve, amennyiben szerepel/. Ebben a példában a file MINTA.SUB-ot a következő SUBMIT utasitással hajtjuk végre:

A>SUBMIT MINTA B:FILE1.TXT↵

A SUBMIT először megkeresi a MINTA.SUB-ot és azután végrehajtja a benne található parancsokat.

A DIR végrehajtásakor a $1 helyébe B-t, a $2 helyébe FILE1.TXT-t helyettesiti be és a file nevét a B lemez directory-jában keresi meg. Azután végrehajtja a PIP-et átmásolva a B:FILE1.TXT-t az A lemezre, ugyanezzel a névvel.

A SUBMIT utasitás általános alakja:

SUBMIT file-név V1V2V3...

ahol V1 a $1-et, V2 a $2-t, V3 a $3-at, stb. helyettesitő érték mindenhol az adott nevü file-ban.
A SUBMIT elvárja, hogy az adott file .SUB tipusu legyen, igy ezt a kiterjesztést nem kell megadni.

Amikor argumentumokkal /$1, $2, stb./ dolgozunk, használni kell a $ jelet, melyet egész szám követ. Az első argumentumnál a számnak 1-el kell kezdődnie, a másodiknál

2-vel. Mivel az argumentumok jelölésére $ jelet használunk, két $ jelet /$$/ kell biztositani ahhoz, hogy a .SUB file-ban egy normál $ jel legyen. /Két $ jel eggyé válik egy .SUB file-ban, mig egy $ jel számmal együtt argumentummá válik./ Egy .SUB file-ban a CTRL--gomb jelölésére a felfelé mutató nyilat használjuk /azaz ↑C vagy ↑Z/.
Ez azért szükséges, mert a legtöbb szerkesztőprogram használata során nem lehet beütni bármilyen CTRL sorozatot a .SUB file készitése közben.

Függetlenül attól, hogy milyen lemezegységet választunk a SUBMIT müvelet végrehajtására, a .SUB file-t nem lehet addig feldolgozni, ameddig a lemezt az A lemezegységbe be nem helyezzük és ujra nem töltjük /melegindités/ a rendszert /CTRL-C elegendő/. Egy folyamatban lévő SUBMIT müvelet a RUBOUT /Delete-törlés/-billentyü beütésével törölhető. A SUBMIT automatikusan létrehoz egy ideiglenes, $$$.SUB file-t, amely tartalmazza a .SUB file parancsait; ez a file kitörlődik a SUBMIT befejeződésekor, vagy ha a rendszer hibát fedez fel a SUBMIT müvelet közben, vagy ha RUBOUT /Delete/ beütésével a SUBMIT müveletet töröljük. Ha valamilyen oknál fogva ez a $$$.SUB file hiba előfordulása után még mindig létezik, bármely rendszer ujratöltés /melegindités/ végrehajtatja a rendszerrel a $$$.SUB parancsait, ahelyett, hogy a begépelt utasitást várná. Ha ez történik, töröljük a SUBMIT müveletet /RUBOUT beütésével/ és a $$$.SUB file-t.

Bármely uj SUBMIT utasitás felülirja a létező $$$.SUB file-t. Természetesen belerakhatjuk a SUBMIT utasitást a .SUB file-ba is, mint egy másik CP/M utasitást.

SUBMIT XSUB-bal

A CP/M 1.4 változatában a SUBMIT egy $$$.SUB ideiglenes file-t hoz létre a .SUB file-ből, s akkor a CCP végrehajtja a $$$.SUB sorait, mintha begépelt utasitások lennének. /A CCP, a Console Command Processor a rendszer azon része, amely a begépelt sorokat beolvassa és gondoskodik azok végrehajtásáról./

A CP/M 2.2 változatában az XSUB program /tranziens program/ egy további lehetőséget jelent. Az XSUB program lehetővé teszi, hogy olyan parancsokat tegyünk egy .SUB file-ba, amelyeknek nemcsak argumentumaik, hanem olyan inputjaik is vannak,mint pl.az ED szerkesztési utasitásai. Olyan inputok fordulhatnak elő, amelyek a CP/M pufferelt input müveletét használják fel. Nem fontos ismerni a pufferelt input használatát, csak annyit kell tudni, hogy a program, amelyet végre akarunk hajtani, s amely részére inputot akarunk biztositani, használ-e pufferelt inputot.

Az ED, DDT és PIP programok mind használnak. Ezért pl. az XSUB segitségével az ED parancsot egy .SUB file-ban paraméterezhetjük is, ugy, hogy a paraméterek aktuális értékét is a .SUB file-ban közöljük.
Nézzünk példát egy ilyen müveletre:

A MUTAT.SUB file tartalmazza a következő sorokat, beleértve az XSUB-ot;

XSUB	XSUB végrehajtás
DIR $1.x	Megjeleniti a file-okat
ED $1.$2	Használja az ED-t a meghatározott file-on

#A	Egy ED beolvasással feltölti a szerkesztési puffert
B	Menjen a puffer elejére
Copyright 1984, LSI	Beteszi a copyright jelzést
E	Befejezi az ED-t
PIP $1.OLD=$1.BAK	Lemásolja a régi backup file-t, amikor az ED befejeződik, a CP/M rendszer visszatér
DIR $1.*	Megjeleniti ujból az összes file-t.

A>SUBMIT MUTAT MINTA.TXT↵

Ebben a példában a SUBMIT először utasitja a CCP-t XSUB végrehajtására. Az XSUB program áthelyeződik közvetlenül a CCP alatti területre és aktiv marad a legközelebbi hideginditásig /rendszer reset vagy hideg inditás/. Amig XSUB müködik, addig az XSUB ACTIVE üzenet megjelenik a rendszer-prompt fölött, s ameddig vannak utasitások a MUTAT.SUB file-ban, az XSUB végrehajtja azokat /hacsak közbe nem avatkozunk egy "CTRL-C"-vel. Esetünkben az XSUB végrehajtja a DIR utasitást, s megjeleniti a MINTA-hoz kapcsolódó file-okat. A MINTA a $1 helyettesitési értéke, a .TXT a $2-és a MUTAT.SUB-ban. A XSUB az ED programot futtatja a MINTA.TXT file-on. XSUB folytatja az input biztositását az ED programhoz /pufferelt konzol input-ot használva/; először a A utasitást használja az ED, ezzel beolvassa a teljes file-t /65335 sor/ a szerkesztési puffer-ba, azután a B utasitja az ED karakter-pointerét, hogy menjen a puffer elejére, ezt követően az I utasitás szerint a "Copyright 1984 LSI" szöveget a puffer elejére kell beszurni, s végül az E utasitással

elmenti az eredményt és befejezi az ED programot. /Az ED utasitások a III. rész 2. fejezetében találhatók/.

Az XSUB program aktiv marad; az ED befejezése után végrehajtja a PIP utasitást: a MINTA.BAK-ot egy uj file-ba, a MINTA.OLD file-ba másolja /elkészitve a MINTA.BAK másolati példányát/. Az XSUB ezután végrehajtja a DIR utasitást a MINTA névvel kapcsolatos összes file nevét megjelenitve a képernyőn.

Amikor a SUBMIT file-ban már nincs több utasitás, a CCP lép a helyébe és vár arra, hogy további utasitásokat gépeljünk a terminálon. Ennek ellenére az XSUB program aktiv marad /a XSUB ACTIVE üzenet az egyes utasitások végrehajtása után is ujra előjön/, hacsak egy olyan programot nem hajtunk végre, amely szándékosan átirja az XSUB által elfoglalt területet /azaz közvetlenül a CCP alatt hideginditást vagy rendszer reset-et végzünk/. Amig XSUB aktiv, a SUBMIT utasitás felhasználhatja egyéb .SUB file-ok végrehajtására. A többi .SUB file-oknak nem kell tartalmazniuk XSUB-ot, mint utasitássort, ha az XSUB már eleve aktiv. Ezért létrehozhatunk .SUB file-okat XSUB nélkül is.

6.4. PIP

A CP/M PIP tranziens parancsa file-ok perifériák közötti átvitelére, ill. file-ok másolására használható. Egyike a leggyakrabban alkalmazott parancsoknak; a lemezes file-okkal kapcsolatos funkciói /melyek egyben legszokásosabb felhasználási területét alkotják/ mellett számos egyéb periféria közötti átvitelre is alkalmazható,

mint például egy lemezes file tartalmának a listázó egységre küldése, vagy egy file olyan adatokkal történő feltöltése, melyek valamely speciális input-perifériáról, esetleg egy modem-en keresztül másik számitógépről érkeznek, stb. A PIP fő felhasználási területét mégis a lemezes file-okkal kapcsolatos müveletek alkotják.

A PIP parancs kiadható ugy, hogy a parancs neve mellett rögtön megadunk parancs-sort is; ez esetben a parancs végrehajtása után a rendszer-prompt tér vissza. Ha viszont nem adunk argumentumot, a PIP saját "*" prompt-jával jelzi, hogy kész parancs-sorok egymást követő fogadására és végrehajtására mindaddig, mig a prompt-ra parancs-sor megadása nélkül, CR leütésével válaszolunk, ekkor visszatér az "A>" rendszer-prompt. Ez utóbbi üzemmódot akkor érdemes használnunk, ha egymás után több file-tranzakciót kivánunk elvégezni.

A PIP parancs-sorának általános alakja lemezes file-ok esetén:

D: cél file-hivatkozás = D:forrás file-hivatkozás

ahol D: a lemezegység/ek/ azonositója/i/, mig a "cél" és a "forrás" szavak jelölik a másolás onputját, ill. inputját. A parancs-sor egyes elemeinek konkrét megadását, lehetséges értékeit, a megengedett elhagyásokat, és a parancs hatására végrehajtott müveleteket részletesen a III. részben a PIP-nek szentelt külön fejezet ismerteti.

6.5. ED

Az ED a CP/M rendszer szerkesztő /editor/ programja. Mint minden ilyen program, un. szöveg file-ok /text file-ok/, tehát ASCII-karaktereket tartalmazó file-ok tetszőleges tartalommal történő feltöltésére, már létező file-ok tartalmának módositására alkalmazható.

Sor-orientált szerkesztő-program, ami annyit jelent, hogy az általa kezelt legnagyobb adategység a sor, tehát nem lehetséges a szöveg ennél nagyobb egységeire /bekezdések, lapok, fejezetek/ hivatkozni, azokat összefüggő részekként kezelve mozgatni, manipulálni. Ez - összehasonlitva a kifinomultabb, tágabb lehetőségekkel rendelkező szöveg-szerkesztő programokkal /word processor/ - az ED-vel történő munkát némiképp nehézkessé teszi.

Az ED által nyujtott fontosabb szolgáltatások az alábbiak:

- még nem létező szöveg-file létrehozása;
- már létező szöveg file szerkesztése az eredeti forrás-szöveg megtartásával;
- kilépés a szerkesztő menetből olymódon, hogy csak a kiindulási helyzet marad meg;
- szerkesztés közben a már megtörtént módositások megőrzése, a már módositott file további szerkesztése;
- szerkesztés közben visszatérés az eredeti állapothoz az időközbeni módositások "elfelejtésével";

- relativ helyzetünkkel azonositott sorok, karakterek törlése; sorok, karakterek megadott helyre történő beszurása;
- megadott szöveg megkeresése és/vagy adott szövegre történő lecserélése;
- más szöveg file-ok tartalmának "beollózása" a szerkesztett file megadott poziciójába;
- utasitások ismételt végrehajtása.

Az ED program utasitásait, azok használatát a III. rész 2. fejezete ismerteti részletesen.

6.6. ASM

Az ASM tranziens parancs az assembly-nyelvű programozás támogatását szolgálja: assembly-nyelvü programokból a mikroszámitógép által végrehajtható hexadeximális gépi kódu programot generál, melyet ezt követően a LOAD parancs segitségével akár be is lehet tölteni a rendszerbe uj tranziens parancsként. Az ASM a forditás közben opcionálisan listát is készit, amely az assembly-nyelvü forrássorokat és a nekik megfelelő gépi kódot tartalmazza. Mivel az assenbly-nyelvü programozással a III. részben külön fejezet foglalkozik, a parancs részletes ismertetésétől itt eltekintünk.

6.7. DDT

A program neve a Dynamic Debugging Tool /dinamikus hibakereső program/ kifejezés kezdőbetüiből adódik. Szintén

az assembly-nyelvü programozás segédeszköze:programok ellenőrzött körülmények közötti próbafuttatására, tesztelésére, hibakeresésre, memóriába történő betöltésére és módositására alkalmas.
Részletes ismertetését lásd a III. rész assembly-nyelvü programozásról szóló fejezetében.

6.8. LOAD

A LOAD parancs egy Intel hexadecimális formátumban lévő /.HEX kiterjesztésü/ file-ban tárolt gépi kódu programból .COM kiterjesztésü file-t készit, melyet ezután a rendszer tranziens parancsaként lehet használni. Alkalmazásának leggyakoribb esete, amikor az ASM paranccsal kombinálva, azt követően aktivizáljuk. Mindez az assembler-nyelvü programozás témakörébe tartozik, igy a parancs ismertetése is ott, a III. rész megfelelő fejezetében található.

6.9. DUMP

A DUMP program tetszőleges, lemezen tárolt file tartalmának az operátori display-n történő megjelenitésére szolgál. A file tartalmát hexadecimális értelmezés szerint listázza; ebből adódóan elsősorban az assembly-nyelvü programozás segédeszközeként használható. Leirását lásd a III. rész 3. fejezetében.

6.10. MOVCPM

A MOVCPM program a CP/M rendszernek az eredetitől eltérő memóriaméretheż történő ujra-konfigurálására alkalmas. Mind a felhasználó által megadott, mind pedig a fogadó mikroszámitógép /host-mikrogép/ RAM-területe által definiált memóriaméretre képes elvégezni a rendszer rekonfigurálását. Szerepe szorosan kötődik a rendszergenerálásħoz, részletes leirása ennek megfelelően az V/4. fejezetben található.

III. RÉSZ:

FILE-KEZELÉS
SZERKESZTÉS CP/M ALATT
PROGRAMOZÁS

1. FILE-KEZELÉS CP/M ALATT: PIP

1.1. PIP funkciók

A PIP a Peripheral Interchange Program /kb. perifériák közti adatcserét megvalósitó program/ röviditése, számos funkciót lát el a CP/M operációs rendszerben:

- Egyetlen file átmásolása egyik lemezről a másikra,
- Egy file másolatának létrehozása az eredetitől eltérő néven,
- Több file átmásolása egyik lemezről a másikra,
- Az összes file átmásolása egyik lemezről a másikra,
- Több file egyesitése egyetlen file-lá /konkatenáció/
- File egy részének másolása,
- Másolás csak olvasható /Read-Only, R/O / file-ra,
- A másolt file tartalmának megjelenitése,
- Másolás rendszer-file-ból,

A fenti lemezes átviteleken kivül a PIP képes file-okat "másolni" egyéb eszközökre pl. nyomtatóra; valamilyen eszközről /pl. papirszalag-olvasó/ file-okat lemezre átvinni, ill. eszközök között is képes adatátvitelt végrehajtani:

- File tartalmának kiküldése valamilyen eszközre /pl. nyomtató/,
- File tartalmának beolvasása valamilyen eszközről /pl. modem/,

- Adatok küldése egyik eszközről a másikra,
- File-ok formattált kiiratása nyomtatóra,
- Valamilyen input eszközről érkező adatok kiiratása a konzolra,
- Kisbetük konvertálása nagybetükké és megforditva,
- Stb.

A PIP-et kétféle módon lehet végrehajtani:

1. egyetlen sorból álló parancsként
2. "programként", amikoris számos parancs-sor megadható egymás után.

1. A PIP parancs-sorának általános formája az alábbi:

d:cél=d:forrás-1,forrás-2,forrás-3...[paraméterek]

ahol a cél az uj file vagy a célperiféria neve, forrás-1, forrás-2, stb. a régi file vagy a forrás periféria neve, d pedig a lemezegység-azonositó./A file neve itt kiterjesztéssel együtt értendő./ A lemezegység lehet ugyanaz mindkét oldalon, de el is térhet egymástól. A jobb oldali d paraméter el is hagyható, mivel ez esetben a PIP a kurrens-lemezegységen keresi a megadott file-t. A bal oldali d is elhagyható, ha megadjuk az uj file nevét, ekkor a másolat a jobb oldalon megadott lemezegységre kerül, vagy a kurrens lemezegységre, ha a jobb oldalon sem szerepel a d paraméter. Az uj file nevét és kiterjesztését sem kötelező megadni, ekkor az uj file-é is ugyanaz lesz, mint a régié. Ez azonban csak akkor alkalmazható, ha a két különböző lemezegység szerepel a

parancs-sor két oldalán, ugyanazon a lemezen ugyanis nem lehet két különböző, de azonos nevü és kiterjesztésü file. A paraméterek ismertetésére a későbbiekben kerül sor.

Az egyetlen soros PIP parancsra példa az alábbi:

```
A>PIP B:LETTER2.DOC=LETTER1.DOC↵
A>
```

Feltételezzük, hogy a forrás file /LETTER1.DOC/ a kurrens lemezegységen van. A parancs hatására létrejön ennek a file-nak egy másolata a B lemezen, LETTER2.DOC néven. Végrehajtás után a CP/M visszaadja a rendszer-promptot /A>/.

2. Ha egyszerre több müveletet akarunk végrehajtani a PIP-pel, akkor programként hivjuk be, ugy, hogy a PIP parancs begépelése után közvetlenül CR-t ütünk, amelynek hatására a PIP bekerül a memóriába, és bejelentkezik a saját promptjával, a *-gal:

```
A>PIP↵
*B:LETTER2.DOC=LETTER1.DOC↵
*A:=B:GAME.BAS↵
*A:=B:NUMBER.FOR↵
*↵
A>
```

Az első parancs-sor megegyezik az előző példával, hatása is ugyanaz. A második és a harmadik sor hatására a

B egységben lévő két file /GAME.BAS és NUMBER.FOR/ átmásolódik az A lemezegységre, az eredetivel megegyező néven. /Természetesen a B egységen is megmaradnak a file-ok./

Több file másolása

Több file másolására az előzőekben bemutatott 2. módszer alkalmas, amikor a PIP-et programként hajtjuk végre. Megkönnyiti az eljárást a joker-karakterek /? és *x*/ használata. A ? helyettesithet egyetlen tetszőleges karaktert. Ha pl. három különböző file neve:

GAME1.BAS
GAMEQ.BAS
GAMEY.BAS

akkor a következő parancs egyszerre elvégzi mindhárom file másolását:

A>PIP A:=B:GAME ?.BAS↵

Ha a B lemez directoryjában szerepel még egy GAMEZ.BAS nevü file is, a parancs hatására természetesen az is átmásolódik.

A *x* joker-karakter tetszőleges hosszuságu karaktersorozatot helyettesithet. Ha pl. létezik az előbb felsoroltakon kivül a GAMEO.COM nevü file is, akkor a GAME?.*x* névvel mind az öt file-ra hivatkozhatunk. Az alábbi parancs

A>PIP B:=*x*.TXT↵

elvégzi az összes, .TXT kiterjesztésü file másolását a B lemezre.

A lemezen lévő összes file másolása a következő módon történhet:

A> PIP B:=*.* ↵

Ezzel a módszerrel csak a directory-ban megjelenő file-ok másolhatók, maga a CP/M rendszer viszont nem /arra a SYSGEN alkalmazása ad lehetőséget/. Irhatunk olyan segédprogramot is, amely teljes lemezek gyors másolását teszi lehetővé /COPY/. Ezzel a CP/M másolása is megtörténik a SYSGEN használata nélkül. A PIP alkalmazásának előnye, hogy a file-nak a lemezen fizikailag szétszórt - nem feltétlenül egymás melletti file-területekben /blokkokban/ levő - részeit egymás melletti területekbe viszi át, lényegesen meggyorsitva ezáltal a file későbbi elérését.

A CP/M 2.2 verziójában több felhasználói terület definiálható. /A felhasználói területekre vonatkozó rövid ismertetést lásd a fejezetben később, a PIP-paraméterek ismertetése után./ Ez esetben a *.* hivatkozás csak az adott felhasználói terület és nem az egész lemez összes file-ját helyettesiti.

Több file egyesitése egyetlen file-lá /konkatenáció/

File-ok összemásolására legegyszerübb példa, ha két file-ból akarunk egyetlen ujat létrehozni:

A> PIP NEWFILE.TXT=FILE 1.TXT,FILE 2.TXT ↵

A jobb oldalon vesszővel elválasztva kell felsorolni az egyesitendő file-ok nevét, ezek száma kettőnél több is lehet. Az összemásolandó file-ok különböző lemezeken is lehetnek, ekkor azonban a lemezegység-azonositót mindig meg kell adni:

A>PIP B:NEW.ASM=A:OLD.ASM,B:OLD1.ASM,A:OLD2.ASM↵

Ha nem adunk meg paramétert, a PIP feltételezi, hogy az egyesitendő file-ok ASCII karaktereket tartalmaznak /szöveg file-ok/. Az ASCII file-ok végét mindig CTRL-Z karakter jelzi, a PIP csak az utolsó file végén hagyja ezt meg, a többiből eltávolitja.

Ha nem szöveg tipusu file-okat akarunk egyesiteni, akkor meg kell adni egy paramétert szögletes zárójelek között, amelyből a CP/M tudni fogja, hogy nem CTRL-Z jelöli a file végét. Részletesebben lásd a PIP paramétereit leiró részben.

File-ok másolása perifériákról és perifériákra

Mint a fejezet elején már emlitettük, a PIP nemcsak lemezek közötti átvitel végrehajtására képes, hanem perifériákra is tud küldeni, vagy onnan fogadni adatokat. Az adatátvitel irányát tekintve a perifériák csoportosithatók bemeneti /input/, kimeneti /output/, illetve be- és kimeneti /input-output/ perifériákra. A papirszalag-olvasók például bemeneti perifériák, mivel csak adatküldésre képesek a központi egység /processzor/ felé. Hasonlóan a nyomtatók kiviteli, a billenytüzettel ellátott videoterminálok /display-k/ be-kimeneti

perifériák. E tulajdonságokat tekintetbe kell venni a PIP parancsok irásakor. /Nem lehet pl. nyomtatóról beolvasni file-okat./

A CP/M operációs rendszer a perifériák jelölésére logikai és fizikai egység-neveket használ. A logikai egység-nevekhez fizikai egységnevek vannak hozzárendelve, a hozzárendeléseket a STAT paranccsal lehet kiiratni, ill. megváltoztatni.

A PIP parancsokban megengedett logikai eszköznevek:

CON: konzol /billentyüzettel ellátott videoterminál/ /be-és kimeneti/

RDR: olvasó /papirszalag vagy kártyaolvasó, de lehet egyéb adatbeviteli eszköz is/ /csak bemeneti/

PUN: papirszalag- vagy kártya-lyukasztó vagy egyéb kiviteli periféria /csak kimeneti/

LST: listakészitésre alkalmas periféria /pl. nyomtató/ /csak kimeneti/

Megjegyzés: a kettőspont karakternek /:/ minden név végén szerepelnie kell.

Néhány példa a fenti logikai egységnevek használatára:

A>PIP↵

*CON:=HELP.TXT↵

* LST:=B:LETTER.DOC↵

*GAME.BAS=RDR:↵

*PUN:=GAME.BAS↵

Az első parancs a HELP.TXT nevü szöveg file tartalmát iratja ki a CON: szimbolikus nevü eszközre /amelyhez valószinüleg a videoterminál van hozzárendelve/.

A második parancs-sor az LST: logikai nevü perifériára /amely általában sornyomtató vagy irógép/ küldi ki a LETTER.DOC nevü file-t, vagyis kinyomtatja azt.

A harmadik PIP parancs beolvas egy file-t az RDR: nevü eszközről /amely papirszalag- vagy kártyaolvasó, esetleg kazettás adatrögzitő/ és lerakja a lemezre GAME.BAS néven.

Ez utóbbi müvelet forditottja érhető el a negyedik parancs megadásával: a GAME.BAS nevü file küldhető ki a PUN: nevü eszközre, amely papirszalag-lyukasztó vagy kazettás adatrögzitő.

A logikai egységnevek mellett fizikai egységnevek is alkalmazhatók a PIP parancsokban:

TTY: konzol, olvasó, lyukasztó vagy listázó

CRT: konzol vagy listázó

PTR: papirszalag- vagy kártyaolvasó

PTP: papirszalag- vagy kártyalyukasztó

LPT: listázó /nyomtató/

UC1: felhasználó által definiált konzol

UR1: felhasználó által definiált olvasó

UR2: felhasználó által definiált második olvasó

UP1: felhasználó által definiált kimeneti periféria

UP2: felhasználó által definiált második kimeneti periféria

UL1: felhasználó által definiált listázó eszköz.

A BAT: fizikai eszköznév nem szerepel a felsorolásban, mivel ez csak azt jelzi, hogy az RDR: és az LST: eszközöket kell a konzol, mint bemeneti/kimeneti periféria helyett használni.

1.2. PIP kulcsszavak

A PIP parancsokban további, speciális eszközneveket is lehet használni:

NUL: 4Ø null karaktert/hexa ØØ-t/ küld a perifériára - amely általában papirszalag-lyukasztó - kb. 1Ø cm-es lyuk nélküli szalagrészt hozva létre. A

```
* PUN:=PROG.HEX,NUL:↵
```

PIP parancs a PROG.HEX nevü file után még negyven null karaktert küld a PUN: logikai nevü perifériára.

EOF: Egy end-of-file /file-vége/ karaktert /ASCII CTRL-Z/ küld a perifériára. /A PIP mindig automatikusan kiküldi szöveg file-ok másolásakor./ PL.:

```
* PUN:=NUL:,PROG.ASM,EOF:,NUL:↵
```

hatására 4Ø null karakter, a PROG.ASM nevü file, egy CTRL-Z karakter és ujabb 4Ø null karakter megy ki a PUN: nevü perifériára.

PRN: Az LST: eszközhöz hasonló, de a nyomtatási kép attól eltérő. HA TAB-karaktert talál, annyi szóköz karakterrel helyettesiti azt, hogy a legközelebbi, 8-cal osztható pozicióba kerüljön a következő nyomtatandó karakter. A sorok 1-től fölfelé számozódnak, a file elején és minden 6Ø sor kinyomtatása után lapdobás történik.

INP: A felhasználó által megirt bemeneti rutin hivása. A felhasználónak a gépi kódra leforditott programot be kell épitenie a PIP-be /"patching"/. Ha az INP: eszköznév szerepel a PIP parancs-sorban, a PIP egy CALL 1Ø3H utasitás végrehajtásával adja a vezérlést erre a szubrutinra. A szubrutinnak a 1Ø9H cimre kell letennie a beolvasott karaktert, a paritás-bitnek nullának kell lennie.

OUT: Az INP:-hez hasonló, de kimeneti eszköznév. Szintén a PIP-be kell beépitenie a felhasználónak saját, az adatátvitelt lebonyolitó szubrutinját. A PIP a C regiszterben adja át a kiviendő adatot, és egy CALL 1Ø6H utasitás végrehajtásával adja a vezérlést a felhasználó szubrutinjára.

Megjegyzés: a PIP memóriaképében a 1ØAH-1FFH memóriahelyek nem használatosak, a felhasználó ide helyezheti el bemeneti és kimeneti szubrutinjait, amelyeket az INP: és az OUT: használ.

1.3. A PIP parancsokban alkalmazható paraméterek

A paramétereket a PIP-ben a file-név után kell irni, szögletes zárójelek közé. Bár az átlagos felhasználó csak néhányat használ közülük, fontos tudnia a többi létezéséről is. A paraméter mindig az előtte levő file átvitelére van hatással, egy zárójelpárban több paraméter is megadható.

[B]

Blokkos átvitel előirása.

Blokk módban a PIP a forrásból a pufferterületre mindaddig viszi át az egymás utáni adatokat, amig egy hexadecimális 13 karaktert /CTRL-S vagy DC3/ nem talál az átviendő adatok között. Miután ezt a karaktert vette, a PIP kiküldi az összes eddig beérkezett, a pufferban lévő adatot a cél file-ba vagy a célperifériára. Az átvitel befejezése után ujra kezdődik a kiürült puffer töltése. A [B]paramétert akkor kell használni, ha a forrás-periféria folyamatosan küldi az adatokat, és ezáltal betelne a pufferterület, ha azt nem üritené ki periodikusan a PIP. Normál, file-ok közötti átvitel esetén nincs szükség a [B]paraméter alkalmazására.

A [B]paramétert általában akkor kell használni, ha papirszalagról vagy kazettás adatrögzitő egységről olvasunk adatokat egy file-ba. Az olvasónak elég gyakran kell DC3 karaktereket beszurnia az adatfolyamba ahhoz,

hogy a pufferterület ne teljen be. Pl.:

*EDIT.ASM=RDR: [B]↵

[Dn]

Az n-edik pozició utáni összes karakter törlése, n 1 és 255 közé eshet, és közvetlenül a D után következik, pl D2Ø . Ez a paraméter csak sor-orientált adatok átvitelekor alkalmazható, amelyekben periodikusan CR /CARRIAGE RETURN-kocsi vissza/ vagy LF /LINE FEED-sor-emelés/ karakterek vannak. Ha a PIP CR vagy LF karaktert talál az átvitel során, attól kezdve elkezdi átvinni a többi beérkező karaktert, egyben meg is számlálja azokat. Az n-edik karakter elérése után a PIP a következő CR vagy LF karakterig nem visz át semmit.

A [Dn] paramétert olyankor használjuk, ha széles /pl. 132 karakteres/ sorokat tartalmazó file-t akarunk megjeleniteni keskeny /pl. 8Ø vagy 4Ø karakteres/ nyomtatón vagy display-n. Használható kommentárok eltávolitására is, ha azok mindig ugyanazon a pozición kezdődnek. File-ok közötti átvitelek esetén ritkán használjuk.

*PRN:=LONG.TXT [D72]↵

[E]

A másolás echózása a konzolon /a másolt adatok megjelenitése a konzolon is/ a másolás során.

[F]

A form feed /lapdobás/ karakterek kiszürése a régi file-ból. Az [F]opció megadásakor a PIP nem másolja át a form feed /hexa ØC, azaz ASCII CTRL-L/ karaktereket. Ezek általában lapdobást idéznek elő, de vannak olyan nyomtatók is, amelyek nem tudják helyesen értelmezni. Ez esetben, vagy display-re történő kiiratáskor lehet szükség a form feed karakterek kiszürésére.

[Gn]

File másolása más felhasználói területekről.

A [G]paraméter lehetővé teszi a PIP számára, hogy a kurrenstől különböző felhasználói területről is másoljon file-okat. n jelzi a felhasználói terület számát, amelyben a forrás-file van, értéke Ø-tól 15-ig terjedhet. Csak a 2.2 verzióban létező paraméter.

[H]

Az átvitt adatok ellenőrzése annak megállapitása céljából, hogy megfelelnek-e az Intel hexadecimális formátumnak /ld. az assembler leirását/. Csak .HEX file-ok esetében használható. Az Intel hexa formátumot általában papirszalag-lyukasztók és -olvasók esetében használják.

Ha perifériáról másolunk file-ba, és kiterjesztésként HEX-et adunk meg, akkor szükségtelen a[H]paraméter

előirása, mivel ekkor a PIP automatikusan ellenőrzi, hogy szabályos Intal hexa formátumuak-e az adatok és hogy helyes-e az ellenőrző összeg /checksum/. Az[I]paraméter azonban ez esetben is alkalmazható.

Ha a PIP hibát észlel az átvitel során, vagyis az adatok nincsenek Intel hexa formátumban, visszaadja a promptot, és vár a hiba kijavitására. Ha lyukszalagot olvasunk be, akkor kb. 5Ø cm-rel vissza kell léptetnünk a szalagot. Ha az olvasó ismét kész /ready/ állapotban van, CR megnyomásával folytatható a szalag beolvasása. /Lásd még az[I]paramétert./

[I]

A null rekordok ignorálása Intel hexa formátumu átvitelek során. A [H] paraméterhez hasonlóan .HEX file-ok átvitelekor használható. Az[I]paraméter előirja a PIP--nek, hogy ne vigye át a null rekordokban levő adatokat /ØØ: az Intel hexa formátumban/. Az[I]megadásakor a PIP automatikusan beállitja a[H]paramétert is. Ily módon a

*PUN:=PROG.HEX [I]↵

parancs ekvivalens a

*PUN:=PROG.HEX [HI]↵

paranccsal.

[L]

A nagybetük konvertálása kisbetükké.
Az A-tól Z-ig terjedő nagybetüket az átvitel során a nekik megfelelő kisbetükké konvertálja.

Az [L]opciót igen óvatosan kell használni. Nem használható olyan file-okkal kapcsolatban, amelyek a számitógépnek szóló parancsokat tartalmaznak.

[N]

Az átvitt sorok számozása az uj file-ban.
Valahányszor a PIP egy CR /CARRIAGE RETURN/ vagy LF /LINE FEED/ karaktert visz át, eggyel növeli a sorszámlálót. [N]paraméter megadása esetén a sorszám vezető nulláit nem irja ki, a sorszám után egy ":" és egy szóköz karakter következik. [N2] esetén kiirja a vezető nullákat, nem tesz kettőspontot, és szóköz helyett TAB-karaktert tesz a sorszám után. Használata file-ok nyomtatásakor hasznos, mivel megkönnyiti a nyomtatási kép olvasását.

[O]

Tárgykódu, vagy más, nem ASCII kódolásu file-ok átvitele. Akkor kell használni, ha nem ASCII kódolásu file-okat /program, tárgykód, bináris adat file-ok/ akarunk átvinni. A .COM kiterjesztésü file-ok másolásakor nem kell megadni, mivel a PIP tudja, hogy a .COM file-ok nem ASCII file-ok.

Az [O]paraméter alkalmazásakor a PIP nem tekinti end of file /file vége/ karakternek a CTRL-Z /hexa 1A/ karaktert, hanem ugyanugy átviszi, mint a többi karaktert. Nem ASCII kódolásu file-ok átvitelét a CP/M akkor fejezi be, ha átvitte az utolsó, az adott file részére a lemezen allokált rekordot. Az átvitel itt tehát nem

végkarakterig tart, mint az ASCII file-ok esetében /CTRL-Z-ig/, hanem a file számára lefoglalt terület nagysága szabja meg az átvitel végét.

[Pn]

Lapdobási /Form feed/ karakter /ØCH/ kibocsátása minden n-edik sor után.
Akkor használjuk, ha a cél eszköz az LST: /általában nyomtató/. Megadásakor a PIP egy lapdobási karaktert szur be az adatfolyamba minden n-edik sor után, ahol n 1 és 255 közé eső decimális szám. Ha n-et nem adunk meg, vagy n=1, akkor minden 6Ø sor után történik lapdobás.

Ha az [F] paraméterrel együtt használjuk, az átvitel alatt detektált összes lapdobási karaktert eltávolitja a file-ból a PIP, és uj lapdobási karaktereket szur be a P paraméternek megfelelően, ujraformázva ezáltal a nyomtatási képet. Azok a nyomtatók, amelyek nem kezelik a lapdobási karaktert, hatására hibásan müködhetnek.

[QstringCTRL-Z]

File egy részének másolása a paraméterben megadott string első előfordulásáig. Hatására a PIP addig folytatja a megadott file másolását, amig meg nem találja a stringet. /Magát a stringet a Q és a CTRL-Z között adjuk meg./

A Q paramétert követő stringet nagybetükké konvertálja a PIP, ha a PIP parancsot a CP/M prompt /A> / után

gépeljük be. Ha a parancsot a PIP prompt-ja /*/ után adjuk meg, akkor ez az automatikus konverzió elmarad. Ily módon, ha kisbetüket is meg akarunk adni a stringben, akkor a parancsot a PIP promptja után kell megadni. Lásd még az [S]paramétert is.

[R]

Rendszer-file /$SYS attributumu file/ másolása. Csak az [R]paraméterrel ellátott PIP parancs képes rendszer file-ok másolására. A rendszer file attributum is átmásolásra kerül. Ez a paraméter csak a 2.2 verzióban létezik.

[SstringCTRL-Z]

File egy részének másolása a paraméterben megadott stringtől kezdődően. A [Q]paraméterhez hasonlóan a file egy részének másolását irja elő. A másolás akkor kezdődik, ha a PIP felismerte a paraméterben megadott stringet /karaktersorozatot/. Az átvitel végét a CTRL-Z detektálása jelenti. Ugyanabban a PIP parancsban szerepelhet [Q] és [S] paraméter is. A kisbetük konvertálására a [Q]paraméter ismertetésekor leirtak érvényesek.

[Tn]

TAB-karakterek helyettesitése szóközökkel.
Hatására a PIP annyi szóközzel helyettesiti az észlelt TAB-karaktereket /hexa Ø9, azaz ASCII CTRL-I/, hogy a következő karakter a következő tabulátor-pozicióra kerüljön. A tabulátorpoziciók távolságát n értéke határozza meg. Alkalmazása olyan nyomtatók és videoterminálok

esetében hasznos, amelyek nem tudják helyesen értelmezni a TAB-karaktereket.

[U]

Kisbetük konvertálása nagybetükké.
A szöveg file-okban előforduló kisbetüket nagybetükké alakitja a PIP az átvitel során e paraméter hatására.

[V]

A másolatként készült file ellenőrzése összehasonlitással.
A másolatnak lemezes file-nak kell lennie; ha periféria, a[V]paramétert figyelmen kivül hagyja a PIP. Az átvitel befejeztével a másolatot ujraolvassa a PIP, és összehasonlitja a memóriapufferban levő eredeti tartalommal.

Több file egyesitésekor csak az első forrás file neve után kell megadni. Használata vitatott, egyes vélemények szerint az általa kimutatott hiba esetén amugyis mindig megjelenik a BDOS ERR ON d: BAD SECTOR hibaüzenet.

[W]

Lehetővé teszi az irást csak olvasható /R/O/ file-okba.

[Z]

A paritásbit törlése az átvitel során.
Perifériáról érkező ASCII kódu adatok vételekor nullázza a legfelső /nyolcadik/ bitet. Az ASCII kód 7 bitet

használ, a nyolcadikat sokszor paritásbitként használják. ASCII kódolásu adatok feldolgozásakor rejtélyes hibát okozhat, ha a paritásbit nem nulla.

Néhány példa a paraméterek alkalmazására:

*LST:=LETTER.TXT [NT8P6ϕ]↵

A fenti parancs kiküldi a LETTER.TXT file-t az LST: eszközre, a nyomtatási kép sorszámozott, a tabulátor-pozíciók minden 8. oszlopban vannak, és minden 6ϕ sor kinyomtatása után lapdobás jön létre. Ez egyenértékü az alábbi paranccsal:

*PRN:=LETTER.TXT↵

A PRN: alapértelmezése megváltoztatható, pl.:

*PRN:=LETTER.TXT [P4ϕ]↵

hatása azonos az előző PIP parancséval, kivéve, hogy a lapdobás nem 6ϕ, hanem 4ϕ soronként történik.

1.4. A PIP uj lehetőségei a 2.2 verzióban

A CP/M operációs rendszer 2.2 verziójában lehetőség van felhasználói területek definiálására, ezek számozása ϕ 1, 2, stb. A lemez állhat egyetlen felhasználói területből is /ϕ-ás terület/, biztositva ezzel a kompatibilitást korábbi CP/M verziókkal. Ha több felhasználói területet használunk, a USER parancs teszi lehetővé az

átjelentkezést az egyik területről a másikra. Ez esetben más felhasználói területre, mint amelyet éppen használunk, nem lehet file-másolatot létrehozni, sőt onnan áthozni sem lehet file-okat a saját felhasználói területre.

A 2.2 verzióban a file-okhoz négy különféle attributum rendelhető, ezek:

- csak olvasható
- olvasható/irható
- rendszer /$SYS jelölésü, a directoryban nem szerepel/
- directory

Ha a lemez állapotát STAT paranccsal lekérdezzük, ezen attributumoknak a képernyőn látható megfelelői rendre:

R/O, R/W, $SYS, DIR.

A PIP-ben három olyan, az előzőekben felsorolt paraméter adható meg a 2.2 verzió alatt, amelyekkel "felülbirálhatók" a felhasználói területek és a file-attributumok:

[Gn] File másolása más felhasználói területekről

[W] Csak olvasható file-ok irásának engedélyezése

[R] $SYS attributumu /rendszer/ file-ok másolása.

Másolás felhasználói területekről

Ha más /nem a kurrens/ felhasználói területről akarunk másolni, akkor a [Gn] paramétert kell használni. Nem kell azonban alkalmazni ezt a paramétert, ha a kurrenssel megegyező sorszámu, de másik lemezegységen levő felhasználói területről akarunk másolni, a másolat azonban

csak a kurrens felhasználói területen /amelybe be vagyunk jelentkezve/ keletkezhet. Pl.

```
*A:=B:HELP.TXT [G3]↵

*↵

2A>
```

Az első sorban begépelt PIP parancs hatására a B lemezegység 3-as felhasználói területén levő HELP.TXT nevü file átmásolódik az A egység 2-es felhasználói területére /ennek sorszámát a PIP kilépése után visszaadott prompt jelzi/.

Ha a CP/M korábbi verzióival kompatibilis rendszert akarunk a 2.2 verzióban, csak a ∅-ás felhasználói területet használjuk. Amennyiben ettől eltérő felhasználói területeink is vannak, a PIP.COM file-t minden olyan felhasználói területre be kell másolni, ahová később file-okat akarunk küldeni. Ha egy lemez összes felhasználói területén van PIP.COM, akkor könnyen elindithatjuk a PIP-et egy másik lemezegységről az előző egység specifikálásával. A PIP-nek az egyes felhasználói területekre történő bemásolását a DDT és a SAVE parancsok használatával az alábbi módon végezhetjük el:

```
A>USER ∅↵

A>DDT PIP.COM↵

DDT VERS.nn.nn

NEXT      PC

1C8∅      xxxx

-G∅↵
```

A>USER 3↵

A>SAVE 28 PIP.COM↵

Feltételezzük, hogy a Ø-ás felhasználói területen már van PIP.CØM, ezért ide jelentkezünk át a USER parancs segitségével. Ezután a DDT segitségével a SAVE parancs leirásakor megismert módon megállapitjuk a PIP hosszát /a NEXT alatt megjelenő cim felső byte-ja hexa 1C, ami decimális 28, ennyi a lapok száma/. Ezután átjelentkezünk a 3-as felhasználói területre, mivel ide akajuk átmásolni a PIP.COM-ot /USER3/, majd a SAVE paranccsal kimentjük a memória 28 lapnyi területének tartalmát, ami éppen a PIP.COM nevü file.

Csak olvasható file-ok

A PIP nem irja felül és nem törli a csak olvasható /R/O attributumu/ file-okat. Ha mégis megpróbáljuk, a PIP a következő üzenetet irja ki:

DESTINATION FILE IS R/O, DELETE /Y/N/?

figyelmeztetve, hogy a cél file csak olvasható, és megkérdezve, hogy törölje-e. Ha Y-t ütünk /↵ nem szükséges/, a régi file-t törli, és létrehozza az uj másolatot, ennek azonban nem lesz automatikusan R/O attributuma. Ha N-et adunk meg válaszként, akkor nem végzi el a másolást a PIP, és a következőt irja ki:

** NOT DELETED **

Ha nem akarjuk, hogy a PIP figyelembe vegye az R/O attributumot, a W paramétert kell megadnunk.

2. SZERKESZTÉS CP/M ALATT: AZ EDITOR /ED/

2.1. Bevezetés

Ebben a fejezetben bemutatjuk a CP/M alatti ED editort /szövegszerkesztőt/. Ismertetjük, hogyan lehet használni, milyen funkciói vannak és hogy a konzol és a memória, ill. a lemez közötti adatmozgások/mozgatások hogyan zajlanak le.

Természetesen részletesen foglalkozunk az editor parancsaival is, példákon bemutatva kivánjuk a megértést elősegiteni.
Voltaképpen mit is nevezünk editornak, mi is egy editor-program?

Valamely editorral un. text file-okat /szöveg file-okat/ hozhatunk létre, azon módositásokat, javitásokat végezhetünk. Az editornak biztositania kell, hogy egyszerüen lehessen editálás közben sorokat, lapokat váltani, a sorbeli karaktereket módositani, lecserélni, lehessen karaktereket törölni, ill. beszurni. Szükséges, hogy lehessen karakter-sorozatokat /string-eket/ kerestetni, sőt ha kell akkor a kérdéses stringet egy másikra le is lehessen cserélni. Az editor fontos sajátossága, hogy az igy kialakitott forrás- vagy szöveg-file-t, az editálás befejeztével lemezre /floppy-lemezre/ kiirathatjuk, és a későbbiek folyamán - ha ismét szükségünk van erre a forrás-file-ra, onnan ujfent beolvashatjuk; az ujonnan szükségessé vált változtatásokat ismételten végrehajthatjuk.

A továbbiakban vizsgáljuk meg, hogyan is működik a CP/M alatt használható ED.COM szövegszerkesztő program.

2.2. Belépés az ED-be

Az ED.COM editor program /a továbbiakban: editor/, mint minden tranziens program, lemezes file-ban helyezkedik el. Behivása az alábbiak szerint történik:

ED x:input-file y:output file

ahol x: annak a lemezegységnek az azonositója, amelyen az input-file van. Ha ez a kurrens lemezegység, megadása nem kötelező, elhagyható;

"input-file" - a szerkesztendő file-t leiró egyedi file-hivatkozás /azaz file-név+kiterjesztés/;

y: az output /cél/ lemezegység azonositója. Ha ez megegyezik azzal az egységgel, amelyről az editort behivtuk, megadása nem kötelező, elhagyható. Ha az "output-file" nem került megadásra, csak az output lemezegység azonositója, a magában álló y:-al azonositott lemezre fog kerülni az input-tal megegyező nevü és kiterjesztésü eredmény file, mig az input-file kiterjesztését .BAK /BAcKup copy-biztonsági másolat/-ra változtatja a szerkesztés befejezésekor a program;

"output-file" - az eredmény-file neve. Ha elhagyjuk,

az output file neve meg fog egyezni az input-éval, mig az eredeti /input/ file kiterjesztését az editálás lezárása után .BAK-ra változtatja az editor.

Az ED-parancsot ne adjuk ki "üresen", azaz az editálandó text-file neve nélkül, mert ez hibát eredményez. Ha "FILE IS READ ONLY" vagy "SYSTEM FILE IS NOT ACCESSIBLE" hibaüzenet jelenik meg a parancs kiadásának hatására, akkor a STAT-parancsot alkalmazhatjuk a hiba elháritására. /Ld. a STAT-parancsot ismertető II/ 6. fejezetet./

Ha az editornak olyan file-t adunk meg, amelyet a directory-ban nem talál meg, akkor feltételezi, hogy uj forrás file-t /source file-t/ kivánunk létrehozni. Ezt a képernyőn megjelenő NEW FILE üzenet jelzi. Ilyenkor a klavitauráról beadott forrás-sorokat fogja fel "source-file"-ként.

Ha olyan nevet adunk be, amely a directory bejegyzései között már szerepel, az editor ebből a file-ból fogja a forrásszöveg-sorokat a memóriába olvasni. A betöltés megtörténtével az editor saját "*" prompt-jával jelzi, hogy készen áll parancsok fogadására.

Példák az editor hivására:

A > ED A:proba.asm B:probal.asm↵

/az A lemezegységről betölti a "proba.asm" file-t, és a B lemezegységre viszi majd ki a "probal.asm" eredmény-file-t./

A>ED proba.pas↵

/az A lemezről betölti a "proba.pas" forrás-file-t, normális lezárás után pedig létrehoz az input-ból egy "BAK" kiterjesztésü file-t /proba.bak/, az outputot pedig szintén az A lemezre fogja tenni, az inputtal megegyező névvel /proba.pas//.

A>ED a:proba.cbl b:↵

/betölti a proba.cbl filet az A lemezről, majd normál lezárás után a B lemezre viszi ki az eredmény file-t, az A-ra pedig a BAK kiterjesztésüt/.

2.3. Az editor parancsairól általában

Mint már volt róla szó, az editor, behivása után saját promptjával jelzi, hogy kész szerkesztési utasitások fogadására:

*

a "*" karakter megjelenése után különböző parancsokat adhatunk ki az editornak, lényegében az ABC betüinek felhasználásával. Bizonyos betük előtt megadhatunk egy /max. 32.768 értékü/ számot; ha nem adunk meg számot, akkor ott 1-et tételez fel. A ∅-nak /nulla/ bizonyos parancsok esetében speciális jelentése van, erre a megfelelő helyeken természetesen felhivjuk a figyelmet.

Némelyik parancs esetében a számértéket még előjel is megelőzheti, ami az éppen kezelt sorhoz és/vagy karakterhez képest mért relativ irányra utal.

Bizonyos parancsok esetében egy karaktersorozatot /string-et/ kell megadnunk; a string lezárása a CTRL-Z karakterrel történik.

Lehetőség van arra is, hogy egy parancssorban több parancsot irjunk le egymás után, elválasztás nélkül. A parancs-sorozat teljes lezárása ebben az esetben is a CR lenyomásával történik.

Az editor a CR-karakter észlelésekor kezdi meg a parancs-sor végrehajtását, a megadás sorrendjében. A parancsok végrehajtását bármikor megszakithatjuk a szóköz /"space"/ billentyü leütésével.

2.4. Az ED müködése

A program a szerkesztést ténylegesen nem a lemezen végzi; a forrás file-t beolvassa a memóriába, pontosabban egy, a memóriában elhelyezkedő pufferbe. A file a pufferbe egyszerre általában nem fér be, ezért, "darabonként" kell a memóriába tölteni. Ezeket a darabokat a felhasználó irja fel a velük történt munka végeztével, később ismertetésre kerülő módon a lemezre egy átmeneti, un. temporary file-ba.

A RAM-memóriában lévő pufferbe beolvasott szöveg tetszés szerint változtatható a forrás-file tartalmának módositása nélkül: a szerkesztés maga mindig a memória-pufferben kerül végrehajtásra. A szerkesztés befejezésekor a memória-puffer és a lemezről még be nem

olvasott forrás file-részek is az editor átmeneti file-jába kerülnek. Ilyenkor az eredeti file-kiterjesztést .BAK-ra lehet változtatni és az uj file az eredeti nevet kaphatja /amennyiben nem adtuk meg explicite az output-file nevét/. Ez lehetővé teszi a felhasználó számára, hogy ne csak az uj verzióhoz férjen hozzá, hanem az előzőhöz is. Hiba esetén vissza lehet térni az előző verzióhoz az ERA és a REN parancsok segitségével. Például beirjuk, hogy

A>ERA SZIG.TXT↵,

hogy kitöröljük a tönkrement programot; az

A>REN SZIG.TXT=SZIG.BAK↵

parancs ezt követő kiadása eredményeképpen a backup lesz ismét az aktuális verzió. /Ezzel az eljárással természetesen megszünt a backup file./

Ha a szerkesztést nem a normál lezárással fejezzük be, mindez elmarad, és a forrás-file tartalma nem módosul.

Ha a file rövid, egyetlen olvasással be lehet olvasni az egész szöveget a memóriapufferbe, ilyenkor nincs szükség a puffer tartalmának időnkénti kiirására az átmeneti file-ba, ill. a puffer uj forrás-sorokkal történő feltöltésére. A memória-pufferbe olvasható sorok száma a sorok feltöltöttségétől és gépünk memóriaméretétől függ, maximális értéke 65535 lehet.

2.5. Forrásszöveg mozgatása az editorral

A forrás-file és a memória-puffer, ill. az utóbbi és az átmeneti file közötti átvitelre használható ED-parancsok az A /Append/ és a W /Write/. Az nA formában megadott parancs a következő n sort a forrás file-ból a memóriába olvassa, az nW parancs az első n sort a memória pufferből az átmeneti file-ba irja, ugyanakkor a memória-pufferban maradt sorokat automatikusan a puffer tetejére tolja.

n egy 1 és 65535 közötti számot jelent. Helyettesithetünk az n helyére egy # jelet, ekkor az n értéke 65535, illetve az összes sor. A parancsnak ez a formája akkor hasznos, ha nem ismerjük pontosan az irandó, vagy olvasandó file hosszát, de biztosak vagyunk abban, hogy belefér a memória pufferbe. Igy a #A beolvassa a teljes file-t a memóriába, a #W beirja a teljes puffer tartalmát az átmeneti file-ba. Az ED hiba-üzenetet küld, ha a puffer-méret mégis kicsinek bizonyul; ilyenkor csak olyan ED paranccsal lehet folytatni a munkát, mely nem növeli a pufferben levő szöveg méretét. Az ED parancsoknál n-t el lehet hagyni, ilyenkor az A és W a szöveg egy sorát olvassa vagy irja. Ha az A parancsnál n = $\emptyset$, az editor a rendelkezésre álló, még üres memóriahelyet félig tölti fel.

Az ED-nek a memória pufferen, a forrás file-on és az átmeneti file-on kivül még egy nagyon fontos jellemzője van, a karakter pointer /karakter mutató/. Az ED a szöveg file-t a szövegsorok szoros egymásutánjaként tekinti, melyet CR-karakter /kocsi-vissza/ választ el

egymástól. Az operátor parancsokkal mozgathatja a karakter pointert a memória pufferben. A karakter pointer /CP/ lehet az első sor első karaktere előtt, az utolsó sor utolsó karaktere után és két karakter között. Egy file szerkesztésénél,egy rossz betü kicserélésénél a karakter pointert a rossz betühöz kell mozgatni. Az összes szerkesztő parancs referencia pontja a CP, ezért nagyon fontos a CP helyének ismerete. A CP mindig a memória-puffer belsejében van és nem lehet a forrás-, vagy az átmeneti file-ba mozgatni.

Van még egy editor-parancs, amellyel a forrásszöveg mozgatni lehet a memória és a lemez között, ez az nX parancs: a CP-től kezdődően n-számu sort ir ki egy X$$$$$$$$.LIB nevü átmeneti file-ba /nem tévesztendő össze az editor átmeneti file-jával, ahová a már megszerkesztett szöveg kerül a memória-pufferből/. Ez a file az ED-ből történő kilépésig, ill., ha nem szabályosan ér véget a szerkesztés /például melegindításssal/, az editor ujra-behivásáig él, akkor azonban törlődik. Ha szükséges, ØX paranccsal szándékosan is törölhetjük ezt az átmeneti file-t. Az X parancs alkalmazása szorosan összefügg az R /Read/ paranccsal, használatát lásd a forráskönyvtárakról szóló 2.9.2. bekezdésben.

2.6. Szöveg beszurás

Az ED-vel történő munka kezdetekor a memória puffer üres. Feltöltése, mint már láttuk, történhet egy létező file beolvasásával, vagy uj szöveg beirásával. Uj szöveg beirására az I /Insert/ parancsot használhatjuk: egy I-t és egy CR-t kell bebillentyüzni. Ennek hatására CP ilyenkor a puffer elején van.

Az ED beszurási üzemmódban tetszőleges beirt sort elfogad és beteszi a memória puffer-ba a beirás sorrendjében mindaddig, mig egy CTRL-Z-vel nem jelezzük, hogy véget akarunk vetni a beszurásnak, és vissza szeretnénk térni ED parancsmódba. A CP az utolsónak beirt karakter utáni helyen marad. Természetesen alkalmazható az I parancs olyankor is, ha már létező forrás file-ba kivánunk beszurásokat eszközölni. A beszurt sor/ok/ ilyenkor is a CP elé kerülnek. Ha a beiktatott szöveg tul sok /a memória-puffer megtelik/, az editor hibaüzenetet ad. A szöveg-beszurás közben a RUBOUT /billentyü/ használható az utolsó karakter törlésére. Nagyobb hibák esetén CTRL-U-val az egész begépelt sort lehet törölni és ujra irni.
Összefoglalva, szövegbeszuráshoz I után egyszerüen beirhatjuk a programot vagy szöveget, a beszurási üzemmódot CTRL-Z-vel lehet befejezni.
Az ED ezután *-gal jelzi, hogy ismét parancs-módban van, uj szerkesztési parancsot vár.

2.7. A szerkesztés befejezése

Az ED-t parancs módban az E /End/ paranccsal lehet lezárni. Ez beirja a szöveget a RAM pufferből az átmeneti file-ba, átmásolja a forrás file maradék sorait is ebbe az átmeneti munka file-ba, és a korábban leirt módon átnevezi a file-okat /a forrás file-t .BAK tipusunak, a munka file-t arra a névre, amit az output-file neveként specifikáltunk, ha azt nem adtuk meg, az output-file neve megegyezik a forrás file-éval/ és átadja a vezérlést a CCP-nek. A következő parancsok szintén alkalmasak az editorból való kilépésre:

H Reedit: Automatikusan végrehajt egy E parancsot, utána visszatér Edit módba, ha nem specifikáltuk az output-file-t; ilyenkor az átmeneti file lesz az uj forrás file, a memória puffer kiürül, és egy uj átmeneti file jön létre. Ha megadtuk az output-file attributumot az ED parancs-sorában, akkor a H parancs hatására az operációs rendszerhez való visszatérés következik be. A H parancs a már megtörtént szerkesztések mentésére alkalmazható.

O Original: Visszatérés az eredeti file-hoz. A memória puffer kiürül, az átmeneti file törlődik, a CP visszaállitódik a file elejére. Ez az előző parancsok hatását megsemmisiti.

Q Quit: Kitörli a megszerkesztett szöveget, a forrás file marad a szerkesztés előtti állapotban, és visszatér a CP/M-hez.

Érdemes megjegyezni, hogy az E, H, O és Q parancsokat csak parancs-módban "*" prompt után begépelve fogadja el a rendszer.

2.8. Elemi szerkesztési parancsok

A pufferben lévő szöveg szerkesztésének fontos tényezője a karakter pointer /CP/. A mozgatásához szükséges parancsok elengedhetetlenek az editor használatánál. A következőkben megvizsgáljuk az alapvető szerkesztési parancsokat:

Müvelet	Parancs	Magyarázat
CP mozgatása a puffer elejére vagy a végére	+/-B	- esetén a puffer elejére, + esetén a végére mozgatja a CP-t.
CP mozgatása karakterenként	+/-nC	n számu karakterrel mozgatja a CP-t, - esetén a puffer eleje, + esetén a vége felé. /a kocsi vissza-soremelést - CR, LF - két külön karakternek veszi/ A memória-puffer határának tullépése esetén az utasitás hatástalan.
Karakter törlés	+/nD	A CP-től kezdve töröl n számu karaktert /- esetén a CP előtti n db-t/.
Sor törlés	+/-nK	Előjeltől függően töröl n sort a CP-től előre vagy hátra számitva. Ha a CP a sor közepén áll, + esetén a CP előtti karakterek maradnak meg, - esetén pedig az utána következők.
CP soronkénti mozgatása	+/-nL	Ha n=∅ a CP-t a sor elejére viszi /ha már eleve nem volt ott/. Ha n≠∅, a CP annak a sornak az elejére megy, amelyik n sorral feljebb /-/, vagy lejjebb /+/ van.

		A CP megáll a memoria puffer tetején, ill. alján, ha n-re tul nagy értéket adunk meg.
Képernyőre irás laponként	+/-nP	n képernyőt, azaz n-szer /általában/ 22 sort listáz ki a display-re. Ha n=ϕ, a CP-től kezdődően listáz ki 22 sort.
Képernyőre irás soronként	+/-nT	A CP-től kezdődően n sort ir ki a képernyőre, értelemszerüen az adott sort követően /+/, ill. megelőzően /-/. Ha n=ϕ, a sornak a CP előtti, ha n=1, a CP mögötti részét jeleniti meg. Ha nem irjuk ki T-t, a parancs jelentése azonos +/-nLT-vel.

A müveletek során definiált n egy ϕ-tól 65535-ig terjedő számot jelent.
Az n=1 értéket nem szükséges megadni. Az n helyébe # jelet helyettesitve n=65535. A +/- egy + vagy egy - előjelet jelent, ha egyik sincs megadva, az editor +-t tételez fel.

2.9. Az editor további lehetőségei

2.9.1. Szöveg keresés és módositás

Az editornak ez a funkció egy nagyon látványos szolgáltatása, mert automatikusan megkeres a szövegben egy szót.

Akkor használjuk, ha a szövegben több helyen előforduló karakter-sorozatot meg szeretnénk változtatni. Erre szolgál az F /Find/ parancs, melynek formája:

nF Szöveg

ahol a Szöveg az a karakter-sorozat, amit keresünk. A parancsot CR-rel vagy CTRL-Z-vel zárhatjuk le.

Az ED a CP-től kezdve átvizsgálja a szöveget, keresi a kivánt karakter-sorozat előfordulási helyét. Ha megtalálja, a CP-t a sorozat utolsó karaktere utáni helyre tolja. A CP nem mozdul abban az esetben, ha a szöveget nem találja meg. A keresés n-szer ismétlődik; ha n-nek nem adunk értéket, automatikusan 1 lesz.
Az ED "##" jelet ir ki és várja a következő utasitásokat, ha a keresett karakter-sorozatot nem találja. Ha a sikertelen F-et tartalmazó sorban több utasitás volt, azok nem hajtódnak végre.

A könnyebb kezelhetőség érdekében az ED-nek létezik az F/Find/ utasitáshoz hasonló másik parancsa, amely a keresés alatt automatikusan irja és olvassa a sorokat.

Formája:

nN Szöveg

A parancs a Szöveg -karaktersorozat n-edik előfordulása utáni karakterre állitja a CP-t. Ha a memóriában az n-edik előfordulást még nem találta meg, akkor ujabb sorokat olvas be az inputból, és a memória

megtelte esetén ugyanakkor sorokat visz ki az outputra. Ha mindezek után sem találta meg az n-edik Szöveg karaktersorozatot, akkor hibaüzenetet küld.

Az editor nemcsak megadott szövegek megkeresését teszi lehetővé, hanem a megtalált karaktersorozattal történő manipulációra is lehetőséget biztosit. Karakterek-sorozat megkeresésére és megadott karaktersorozattal történő egymás mellé illesztésére alkalmazható az editor J /Join/ parancsa, melynek formája:

nJ Szöveg-1 CTRL-Z Szöveg-2 CTRL-Z Szöveg-3 { CTRL-Z / ↵ }

A parancs három karaktersorozatot helyez egymás mögé, a következő módon: megkeresi a Szöveg-1 karaktersorozatot, majd mögé illeszti a Szöveg-2 -t és ezt követően minden karaktert töröl addig, mig meg nem találja Szöveg-3 -at. Mint látható, a parancsot n-szer is végre lehet hajtani.
Ha a Szöveg-3 karaktersorozatot nem találja meg, nem töröl ki semmit sem.

Itt emlitjük meg, hogy a korábban már bemutatott I /Insert/ utasitásnak létezik egy röviditett formája, amely gyakran használatos kisebb szövegváltoztatások végrehajtására:

I Szöveg

ahol Szöveg az utólag beiktatandó karakterek sorozata. A szöveg közvetlenül a CP utáni helyre kerül beszurásra, a CP ezt követően a beiktatott szöveg utáni pozicióba áll be.

Az ED önálló utasitásként biztositja az F és I parancsok kombinációját. Az S /Search/ parancs megkeresi a megadott szöveget és egy másik, uj karakter-sorozatot ir a helyére.

Általános formája:

NS régi szöveg CTRL-Z uj szöveg CTRL-Z

Az S parancsra az ED a CP-től kezdődő memóriarészben megkeresi a régi szöveg -ben megadott karakter-sorozatot, kitörli és a helyébe az uj szöveg -et helyettesiti. A CP az uj szöveg utolsó karaktere utáni pozicióban marad. Ezt a müveletet n-szer hajtja végre. Az ED a ## kiirásával jelzi, ha a régi szöveg n- vagy már korábbi előfordulását nem találja meg a memória-pufferban.

2.9.2. Forráskönyvtárak

Az ED lehetővé teszi, hogy a szövegünkbe a szerkesztési folyamat alatt, R paranccsal a könyvtárból külön file-okat másoljunk be. A parancs formája:

R file-név

ahol file-név a lemezen lévő .LIB -kiterjesztésü file neve. Az ED beolvassa a specifikált file tartalmát a file vége /CTRL-Z/ karakter érzékeléséig és a beolvasott sorokat a memoria-pufferbe irja a CP utáni helyre, az I utasitáshoz hasonlóan.

Pl. az operátor RCIM utasitásának hatására az ED beolvassa a CIM.LIB file-t is, és tartalmát automatikusan beszurja a memória pufferba, a CP mögé.

Ha a file-név nem kerül megadásra /"üres" R parancs/, az X paranccsal korábban létrehozott, X$$$$$$$$.LIB file tartalmát emeli át a memória-pufferbe.

2.9.3. Parancsok ismételt végrehajtása

Az M /Makro/ parancs lehetővé teszi ED parancs-sorozatok ismételt végrehajtását, a CP-től kezdődően.

Az M utasitás általános formája:

nM parancs-sorozat

ahol parancs-sorozat ED parancsokból álló tetszőleges sorozatot jelent, de nem tartalmazhat másik M utasitást. Az ED n-szer hajtja végre az utasitás sorozatot $n > 1$ esetén. Ha $n=\emptyset$, $n=1$, vagy nem adjuk meg, az utasitások végrehajtása mindaddig tart, mig az lehetséges, ill. amig nem következik be hibajelzés /pl. a memória puffer végét F utasitással érjük el/.

Az I-re, S-re stb. végződő parancs-sorozatot a CR mellett CTRL-Z-vel kell lezárni. Példaként iktassuk be a "lemez" szót ötször a file-ba:

5MILEMEZ ↑ZL↵

2.9.4. Emelt és alaphelyzetü billentyüzet

Az U parancsot követően az ED minden bejött adatot emelt helyzetüre /nagybetü/ változtat. Hatására a karakterek emelt helyzetben /nagybetüként/ tárolódnak a pufferben, annak ellenére, hogy a képernyőn a beirt módon jelennek meg. A -U parancs semlegesiti az U parancs hatását.

Az I, F, S, N parancsok megváltoztatják a Szöveg karaktereket emelt helyzetüre abban az esetben, ha a parancs emelt helyzetben, azaz nagybetüvel iródott. Alaphelyzetben, azaz kisbetüvel beirt parancsok esetén ilyen változtatás nem történik.

2.9.5. Utasitáspozicionálás

Van még az editornak néhány olyan lehetősége, amelyek a viszonylag kényelmetlen ED-programot mégis használhatóvá, hatékonnyá teszik. Ezek tulajdonképpen nem önálló parancsok, hanem a parancsok érvényességi intervallumát leiró operátorok.

$$\left\{\begin{matrix} n \\ \leftarrow \end{matrix}\right\} \quad \left.\begin{matrix} n \\ 1 \end{matrix}\right\} \text{ sort } \left\{\begin{matrix} \text{előre-} \\ \text{-hátra} \end{matrix}\right\} \text{ -lép és ezt a sort kiirja a display-re}$$

n:x az n-edik sorra rááll és végrehajtja az "x" parancsot

:mx az aktuális sortól kezdődően, az m-edik sorral bezárólag végrehajtja az "x" parancsot.

n::mx rááll az n-edik sorra, és innen kezdve, egészen az m-edik sorig végrehajtja az "x" parancsot.

2.9.6. Sorszámozás

Az editor V parancsa lehetővé teszi belső /tehát csak az ED használata közben létező/ sorszám generálását. A V parancs kiadását követően minden sor sorszámmal együtt jelenik meg a képernyőn, az alábbi formában:

. nnnnn: xxxxx

ahol nnnnn 1-65535 tartományba eső szám. ØV hatására a "szabad"/"összes" byte-ok számát irja ki /decimálisan/ a képernyőre, az alábbi formában:

free/total - ahol "free" a még szabad memória, "total" pedig a rendelkezésre álló memória-puffer mérete.

2.1Ø. Az ED hibaüzenetei

Ha az ED programon belül hiba lép fel, akkor az editor kiirja a hibát megelőző utolsó karaktert, az alábbi hibajelzések valamelyikének feltüntetésével:

##	Sikertelen karakter-sorozat keresés Az F, S vagy N utasitás nem találja a kivánt sorozatot.
??X	Fel nem ismert X utasitás karakter. Abban az esetben is előfordulhat, ha az E, H, Q vagy O nem egyedül szerepel az utasitás sorban.
ØØ	Az R utasitásban megadott .LIB file-t nem találja.

>> Az F, N, S sorozat tul hosszu, vagy tele van a memória puffer. /A D, K, N, S, W használható a karakterek pufferból való törlésére./

A >>A hibaüzenetet kapjuk abban az esetben, ha egy A/ppend/ utasitás hatására a memória puffer feltöltődik, ilyenkor a CP a puffer végére kerül. /Normális körülmények között az A utasitás a CP-t nem mozditja el/.

A memória puffer feltöltődését az I utasitás alatt a >> X kiirás jelzi, ahol X a kimaradt karakter, ilyenkor a CP a beiktatott szöveg végére kerül.

A FILE ERROR hibaüzenetet kapjuk, ha a lemez feltöltődött, tele van a directory vagy a CP/M valamelyik file-nál hibát talál. Ebben az esetben az ED megáll és visszatér a CCP-hez; eredmény-file nem jön létre, tehát az editálást meg kell ismételnünk.

2.11. Az editor-parancsok összefoglalása

nA — n sort appendál /ØA esetén a puffer felét használja/

B, -B — A karakterpointer file {elejére, végére} állitja

nC — a karakterpointert n karakterrel előbbre állitja

nD — n karaktert töröl, előre irányban

E — az editálási menetet befejezi /normális lezárás/

nFs — megkeresi az "s" karaktersorozat n-edik előfordulását

H az editálást befejezi, a pointert a file elejére állitja

I karakteréket szur be a karakterpointertől kezdődően, mindaddig ameddig CTRL-Z karaktert be nem adunk.

Is a karakterpointertől kezdődően beszurja az "s" karaktersorozatot.

nK a pointertől kezdődően n sort töröl

nL n sorral átállitja a pointert

nMx n-szer végrehajtja a parancs-sorozatot

nNs globális keresés, a file végéig

O az editálást abortálja, és ujrakezdi az eredeti file-lal.

nP a következő n lapot /22-soros/ kilistázza /n=Ø, az aktuális lap/

Q az editálást abbahagyja, az input file-t nem változtatja meg.

Rfn A fn.LIB-et beolvassa, a pointertől kezdődően, a pufferbe

nSxCTRL-Zy a pointertől kezdődően lecseréli az első n előfordulását az "x" karaktersorozatnak, "y" karaktersorozatra

nT kinyomtat/kilistáz n sort

U emelt helyzetü bevitel /nagybetü/

V lehetővé teszi a belső sorszámozást

nW a puffer kezdetétől kezdődően kiir n sort az output file-ba

nX a következő n sort /a pointertől kezdődően/ kiirja az X$$$$$$$$.LIB file-ba

További lehetőségek:

{n / ↵ / -} {n / 1} sort {előre- / hátra-} lép, és ezt a sort kiirja a képernyőre

n:x előrelép n sort, és végrehajtja az "x" parancsot

:mx az "x" parancsot, az aktuális sortól kezdődően az m-edik sorig végrehajtja

n::mx rááll az n-edik sorra, és onnan kezdődően végrehajtja az "x" parancsot, az m-edik sorig.

3. AZ ASSEMBLY NYELVÜ PROGRAMOZÁS TÁMOGATÁSA A CP/M OPERÁCIÓS RENDSZERBEN

3.1. Bevezetés

Az Intel 8Ø8Ø tipusu mikroprocesszor gépi kódu programozásának megkönnyitésére a CP/M operációs rendszerben több program áll rendelkezésre.

Ezek közül a 3.2. fejezetben mutatjuk be az ASM programot, amely az I 8Ø8Ø-as mikroprocesszor assembly-nyelvü programjának forditását végzi. A forditásnál az Intel cég által javasolt mnemonik készletet alkalmazza. A gépi kódu programok hibáinak kideritésére

szolgáló hatékony software hibakereső programot mutat be a 3.3. fejezet: a DDT /Dynamic Debugging Tool/ programot.

A 3.4. fejezetben a gépi kódu programok betöltésére alkalmazott LOAD programot mutatjuk be. Ezen programmal nyilik lehetőségünk a hexadecimális file-ban tárolt programjainkat az operativ memóriába tölteni, un."COM" kiterjesztésü file-ként.

A 3.5. fejezet bemutatja, hogyan történik a programok dumpolása, azaz listázása hexadecimális alakban, a DUMP program segitségével.

3.2. CP/M assembler /ASM/

3.2.1. Bevezetés

A CP/M assembler forditóprogram assembly nyelvü forrásnyelvi szöveget olvas be a lemez-egységről, majd ezt I 8Ø8Ø-as gépi kódra forditja Intel hexa formátumban.
Az assembler a következő módon hivható az operátori konzolról:

ASM filenév

vagy

ASM filenév.kiterjesztés

Mindkét esetben kell, hogy legyen a lemezen egy olyan forrásnyelvi szöveget tartalmazó file, amelynek neve:

file-név.ASM

/A forrásnyelvi szöveget az ED programmal készithetjük el, amely univerzális felhasználásu editor lehetőséget biztosit./

A program bejelentkezése mindkét esetben a következő formában történik

CP/M ASSEMBLER VER n.n.

ahol n.n az aktuális verzió számát jelöli.

Az első parancs eredményeként az assembler program beolvassa a forrásnyelvi file-ban található információt és két output file-t hoz létre a

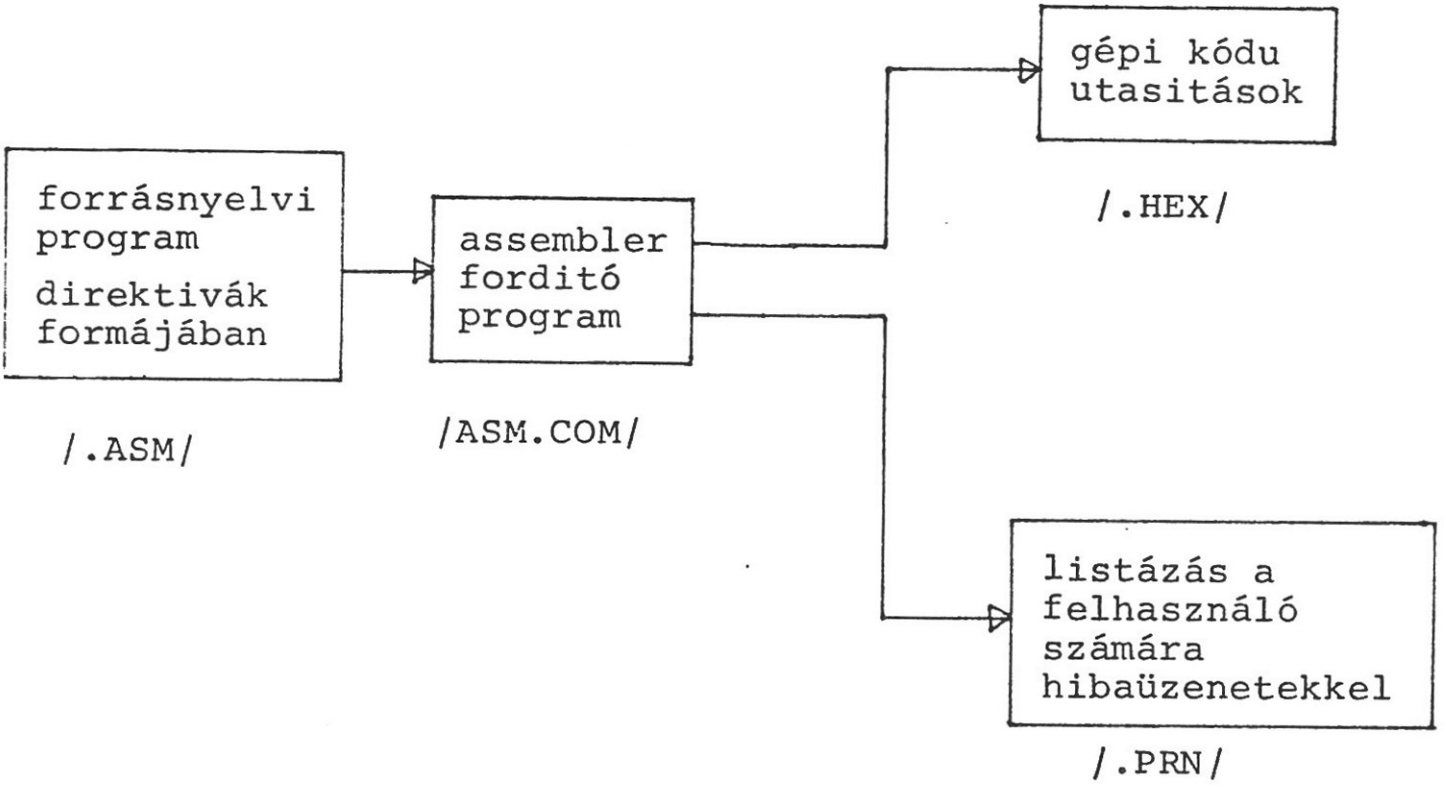
file-név.HEX

és a

file-név.PRN

file-okat.

Az eredmény file-okat arra a lemezegységre helyezi, amelyről a forrásnyelvi file-t beolvasta.



III-1. ábra

A ".HEX" kiterjesztésü file tartalmazza az eredeti forrásnyelvi programhoz tartozó leforditott kódokat Intel hexa formátumban.

A ".PRN" kiterjesztésü file egy olyan kombinált listát tartalmaz, amely az eredeti forrásnyelvi lista soronkénti leforditott hexadecimális kódjaiból, illetve az eredeti forrásnyelvi szövegből áll. Ha a fordító szintaktikai hibát észlel a hibaüzenet kiiródik mind az operátori konzolra, mind pedig a ".PRN" file-ba. A forditási folyamatot a III-1. ábra szemlélteti. Például: a leforditandó forrásnyelvi program

"PROG.ASM"

Ebből az "ASM.COM" fordítóprogram két file-t hoz létre:

a "PROG.HEX"

és a "PROG.PRN" file-okat,

ha a hivás módja például a következő:

A>ASM PROG↵

Az összes file-t az "A" jelü lemezegységen helyezi el.

A második parancsforma felhasználható az input file és az output file-/ok/ helyének, illetve generálásának meghatározására. Ebben az esetben a filenév ".kiterjesztés" része, amely 3 betüből álló betücsoport meghatározza a forrásnyelvi file, illetve az eredmény file/ok/ helyét/lemezegységét/.

Ennek formája a következő:

filenév.p1p2p3

ahol p1, p2 és p3 egy betüs karakterek.

p1 : A,B,.....,P meghatározza a forrásnyelvi programot tartalmazó lemezegység azonositóját.

p2: A,B,.....,P meghatározza a hexa file-t tartalmazó tárgykódu file helyét /lemezegység azonositóját/.

Z nem készit hexa file-t az adott programr

p3: A,B,.....,P meghatározza a lista file-t tartalmazó file helyét /lemezegység azonositóját/.

X a lista file-ba irandó tartalmat az operátori displayre irja ki.

Z nem készit lista file-t a forditás során.

Igy például a következő parancs

A > ASM PROBA.AAA↵

A forrásnyelvi programot az "A" azonositóju lemezegységről veszi és készit hexa /PROBA.HEX/, valamint lista /PROBA.PRN/ file-t, amelyeket szintén az "A" azonositóju lemezegységre helyez.

A parancs hatása ilyen esetben megegyezik a következő parancséval:

A > ASM PROBA↵

A következő utasitás

A > ASM P.CBX↵

a forrásnyelvi programot a "C" nevü lemezegységről veszi és az eredményként előállitott hexa file-t a "B" azonositáju lemezegységre helyezi.

A lista file-t ebben az esetben az operátori display-re küldi és nem készit P.PRN file-t.

Az

A > ASM X.BZZ ↵

esetben a "B" nevü lemezegységről olvassa be a forrásnyelvi szöveget és nem készit sem hexa, sem pedig lista file-t. Ez a forditási parancs a program szintaktikai hibáinak gyors ellenőrzésére szolgál.

A forrásnyelvi program formája megegyezik az Intel 8Ø8Ø assemblerével.
Van azonban néhány olyan speciális utasitás és direktiva, amely a CP/M assemblerben jelentős segitséget nyujt. Ezeket adjuk meg a következőkben.

3.2.2. Az assembler forrásnyelvi programjának formája

Egy forrásnyelvü utasitás elfogadható bemeneti adat az assembler forditóprogram számára, ha a következő elemekből áll:

sorszám / cimke / utasitás/ operandus/ok / megjegyzés

ahol az utasitás sor minden egyes eleme önálló jelentéssel bir.

A forrásnyelvü utasitásokat kocsi vissza soremeléssel /CRLF/, vagy "!" jellel választhatjuk el egymástól.

A sorszám egy decimális egész tipusu szám, amely a forrásnyelvi program sorszámait jelöli.

A cimkemező a következő formáju lehet:

```
        cimke
vagy
        cimke:
```

A cimke jelenléte nem kötelező, azokban az utasitás sorokban alkalmazzuk, amelyeket meg akarunk /pl. vezérlésátadás céljából/ jelölni. A cimke alfabetikus karakterrel kezdődik, ezt alfanumerikus és a $ karakterek követhetik, hossza maximálisan 16 karakter.
Egy ":" követheti, de ez nem kötelező.

A következőkben bemutatott cimkék szintaktikailag helyesek:

```
Z       ry          hosszu$név
K:      ZKl:        meg$ennel$is:
Klf2    Xlx2        Cl23$4567$89Ø:
```

A következők pedig helytelenek:

```
2ablak:     K##12
```

Az utasitásmező egy assembler direktivát, egy pszeudó utasitást, vagy egy I8Ø8Ø gépi kódu utasitást tartalmazhat. A pszeudó utasitásokat és a gépi kódokat a későbbiekben irjuk le.

Az operandus mező konstansokból és cimkékből álló kifejezés, amelyeket aritmetikai, illetve logikai utasitások kapcsolnak össze.

A megjegyzéseket ";" karakter után irjuk a következő valódi vagy logikai sorvégig. Ezeket a karaktereket beolvassa az assembler, a listákon megjeleniti, de a forditás során figyelmen kivül hagyja.

Az assembler forrásnyelvü programot az előzőekben bemutatott programsorok egymásutánjából kell előállitani. A forrásnyelvi programot egy END utasitással kell lezárni. Minden olyan utasitást, amely az END utasitás után következik, az assembler fordító program figyelmen kivül hagy.

3.2.3. Az operandusok formája

Az utasitásokban leirt operációk és pszeudo operációk számára egy olyan forma szükséges, amely az összes utasitás esetén alkalmazható.
Az operandus mező egyszerü cimkék, számkonstansok és utasitásszavak aritmetikai és logikai müveletekkel képzett megengedett kombinációit tartalmazza. Az assemblálás alatt minden kifejezésnek 16 bites értéket kell szolgáltatnia. Ez azt jelenti, hogy egyszerü byte-müveletek esetén is két byte-os adatokkal dolgozik az assembler; ekkor a magasabb helyiértékeket tartalmazó byte nulla értékü.

3.2.3.1. Cimkék

Amint azt a korábbiakban bemutattuk, a cimke egy olyan elem, amely tetszőleges utasitáshoz tartozhat.

Ha a cimke olyan utasításhoz tartozik, amely gépi kódu utasitást hoz létre, vagy memória területet foglal le, akkor a cimke értéke azzal a cimmel egyezik meg, amelyet megjelöl.
Ha a cimke egy EQU vagy SET utasitást jelöl, akkor a cimke értéke az adott utasitás utasitásmezejében lévő operandus értéke.
Például a következő programrészlet esetén:

```
Cim /HEXA/      cimke          utasitás
                START:         EQU     2Ø2ØH
   1ØØØ                        MOV     A,B
   1ØØ1         LOOP:         LXI     H,ØØFFH
                              XOR     C
                              JZ      LOOP
                               .
                               .
                               .
```

a START cimkéhez 2Ø2ØH értéket rendel a forditóprogram, mig a LOOP cimke értéke 1ØØ1H lesz.

Cimkék szerepelhetnek még az operandus mezőben /ahogyan ezt a mintapélda is bemutatja a LOOP cimkével/.
Ebben az esetben az assembler fordító program értéküket az első forditási fázis - direktivák cimkék, változók, konstansok értékelése - után határozza meg.

3.2.3.2. Numerikus konstansok

A numerikus konstansok olyan 16 bites számértékek, amelyeket különböző számrendszerekben irhatunk le. A

számrendszerek jelölése betükkel történik, a számértéket követően. A CP/M assembler a következő számrendszerek használatát engedélyezi:

B	bináris	/2-es alapu/
O	oktális	/8-as alapu/
Q	oktális	/8-as alapu/
D	decimális	/1Ø-es alapu/
H	hexadecímális	/16-os alapu/

A jelölés elmaradása a decimális számrendszert jelenti.

A különböző számrendszerekben leirt számok esetén a hexadecimális számrendszer kivételével csak számjegyeket használunk. A hexadecimális számrendszerben a 1Ø és az ennél nagyobb számok esetén betüjelölést alkalmazunk.

A jelölés a következő:

Hexadecimális	Decimális érték
A	1Ø
B	11
C	12
D	13
E	14
F	15

A hexadecimális számkonstansok és a cimkék megkülönböztetése végett a nem számjeggyel kezdődő hexadecimális számkonstansokat egy Ø /nulla/ számjeggyel kezdjük!

A cimkékhez hasonlóan a számkonstansoknál is alkalmazható a $ jel, amely az olvashatóságot könnyiti meg, de semilyen egyéb értéket nem ad a számkonstanshoz.

A számrendszert jelölő betü, illetve a hexadecimális számkonstansoknál alkalmazott betük egyaránt lehetnek kis és nagy betük.
A következőkben szintaktikailag helyes numerikus számkonstansokat mutatunk be:

1234	1234D	11Ø$ØØ1$B
1234H	ØFEFFH	33$77$22q
3377Q	Øfe3h	1234d

Helytelen számkonstansok:

2.34	4654k

3.2.3.3. Fenntartott szavak

Egyes karaktersorozatok az assembler számára kulcsszavakat jelölnek, igy ezeket nem használhatjuk cimkeként. Ilyenek például az I8Ø8Ø-as mikroprocesszor belső regisztereinek elnevezései. A következő táblázat a lefoglalt cimkéket és a hozzájuk rendelt értéket mutatja be.

A	7
B	Ø
C	1
D	2

E	3
H	4
L	5
M	6
SP	6
PSW	6

Megjegyzés: a kis- és nagybetüvel megadott név itt is ugyanazon értéket jelöli.

A gépikódu utasitások, amelyeket szintén az utasitás mezőben alkalmazunk, a neveikhez rendelt belső kóddal egyeznek meg. Abban az esetben, ha az utasitás operandusokat is igényel, akkor értékeként az operandusok nélküli számértéket tárolja a rendszer. Az utasitás valódi értéke a forditás befejeztével fog kialakulni. Példaként nézzük a MOV A,B utasitást, melynek belső reprezentációja a 78H hexadecimális érték. Ez a következő módon áll elő: Az assembler ebben az esetben a MOV utasitás paraméterek nélküli /zéro értékü paraméterekkel/ értékét tárolja, amely 4ØH értékü. Ezt egésziti ki az assembler esetünkben az A és B regiszterekhez tartozó számértékek az utasitásban meghatározott helyekre történő beillesztésével. Igy alakul ki a MOV A,B utasitáshoz tartozó 78H érték.

A $ utasitást is alkalmazhatjuk az utasitás mezőben, de nem ugy, mint a cimkék, illetve a numerikus konstansok esetében, azaz elválasztójelként, hanem önállóan. Ebben az esetben a $ szimbólum értéke azon utasitás cimével egyezik meg, amelyben alkalmaztuk.

3.2.3.4. Szövegkonstansok

A szövegkonstans aposztróf jelek /'/ között szereplő ASCII karakterek sorozata. Egy szövegkonstáns maximális hossza 64 karakter lehet, amelybe a kezdő és záró aposztróf jel is beleértendő.
Az aposztróf jelek között szereplő karaktereknél nem történik semmilyen transzformáció a kis és nagy betük között. Ez azt jelenti, hogy a kis- és nagybetük egyaránt használhatók.
Szintaktikailag helyes szövegkonstansok a következők:

'A' 'AB' 'ab' 'c'

'Ez egy "HASZNOS" könyv'

'Ez egy név'

Helytelenek

"Ez nem jó" Ez sem

3.2.3.5. Aritmetikai és logikai müveletek

A következő aritmetikai müveletek egy operandusmezőn belül alkalmazhatóak az operandusok /pl. numerikus konstansok, cimkék/ között.

a+b "a" és "b" előjel nélküli aritmetikai összege.

a-b "a" és "b" előjel nélküli aritmetikai különbsége.

+b plusz jel, eredménye b.

-b minusz jel, eredménye Ø-b.

a * b "a" és "b" előjel nélküli szorzata.

a/b "a" és "b" előjel nélküli hányadosa.

a MOD b a maradék a és b elosztása után.

NOT b "b" értékének logikai invere, /minden Ø értékü bit 1-gyé válik és minden 1 értékü Ø-vá/, ahol "b" 16 bites értékü.

a AND b "a" és "b" értékei között bit-ről bit-re logikai "és" kapcsolat.

a OR b "a" és "b" értékei között bit-ről bit-re történő logikai "vagy" kapcsolat.

a XOR b "a" és "b" értékei között bit-ről bit-re történő logikai "kizáró vagy" kapcsolat.

a SHL b "a" értékének "b" helyiértékkel balra történő eltolása. A belépő uj bitek értéke zéró.

a SHR b "a" értékének "b" helyiértékkel jobbra történő eltolása. A belépő uj bitek értéke zéró.

Ahol "a" és "b" egyszerü operandusokat jelölnek, melyek cimkék, numerikus konstansok, fenntartott szavak és egy vagy két karakteres szövegkonstansok.

Ezeken kivül "a" és "b" értékei zárójelezett részkifejezések is lehetnek. Nézzünk a müveletekre néhány példát!

1Ø + 2Ø	1Øh + 37Q	L1/3
(L2+4) SHR 3	('B'+B) OR C	(PSW+M)

Az aritmetikai és logikai müveleteket az assemblálás idején végzi el a forditóprogram, mint 16 bites előjel nélküli müveleteket.
Az eredményül kapott operandus meg kell, hogy feleljen azon müveleti kód követelményeinek, amelynek operandusa. Példaként nézzük a következő utasitást. Ha egy kifejezésben az ADI /add immediate/ utasitást használjuk, az operandusmezőben az utasitás magasabb helyiértékü 8 bitjének zéró értékünek kell lennie.

Az "ADI -1" utasitás esetén azonban hibajelzés történik, mivel az operandusnem 8 bites érték.

/a "-1" érték 16 bites megfelelője ØFFFFH/

Az "ADI (-1) AND ØØFFH" utasitást már elfogadja az assembler, mivel az AND utasitással a magasabb helyiértékü 8 bit értéke zéró lett.

3.2.3.6. Az utasitások sorrendje

Az egyes utasitásokat prioritási sorrendbe állitva elhagyhatjuk a müveletvégzéseknél szükséges zárójeleket.
A következő táblázatban bemutatott utasitások közül elsőként a legmagasabb prioritásu a legutolsóként pedig a legalacsonyabb prioritásu szerepel.

Azon utasitásoknál, amelyek egy sorban szerepelnek és igy azonos prioritásuak, az ugynevezett balról--jobbra szabály érvényes.
Ez azt jelenti, hogy az aritmetikai-logikai kifejezés feldolgozása az olvasási iránynak /balról-jobbra/ megfelelően történik. Az utasitások prioritásának sorrendje tehát a következő:

```
* / MOD  SHL  SHR
    -    +
   NOT
   AND
 OR   XOR
```

A következő mintapéldákban a baloldalon szereplő kifejezések az assembler számára teljesen egyenértékűek a jobb oldalon szereplőkkel.

```
a * b + c              ;    (a * b) + c
a + b * c              ;     a + (b * c)
a MOD b * c SHL d      ;  ((a MOD b) * c) SHL d
a OR b AND NOT c + d SHL e ; a OR (b AND (NOT (c + (d SHL e))))
```

Természetesen a zárójelek alkalmazásával megváltoztathatjuk a müveletvégzések sorrendjét.

3.2.4. Assembler direktivák

Az assembler direktivákat használjuk a cimkék, konstansok értékeinek meghatározására. A direktivák az utasitás mezőben egy un. pszeudo utasitáscsoportot alkotnak.

A CP/M assembler által elfogadott direktivák a következők:

ORG	meghatározza egy program vagy adatmező kezdőcimét
END	a forrásnyelvi program vége
EQU	numerikus konstans deklarációja
SET	numerikus változó deklarációja
IF	a feltételes assemblálás kezdete
END IF	a feltételes assemblálás vége
DB	adatbyte helyfoglalás
DW	adatszó /2 byte/ helyfoglalás
DS	adatterület helyfoglalás.

3.2.4.1. Az ORG direktiva

Az ORG utasitás alakja a következő:

Cimke ORG kifejezés

ahol a "cimke" alkalmazása nem kötelező, a "kifejezés" pedig olyan operandusokat tartalmaz, melyek értékeit az ORG utasitás végrehajtása előtt meghatároztuk. Az assembler azon a cimen kezdi el a gépi kódu program generálását, amelyet a "kifejezés" meghatároz. Tetszőleges szám szerepelhet az utasitásban. Igy a

programozó már felhasznált memóriaterületeket ujra definiálhat és felülirhatja uj utasitásaival /overlapping/.

A CP/M rendszerben a programok száma az ORG utasitás a leggyakrabban a következő alaku:

```
ORG         1ØØH
```

Mivel 1ØØH-n kezdődik az első olyan szabad memóriaterület, ahová a programozó saját programját elhelyezheti /TPA/. Ha "cimkét" alkalmazunk az ORG utasitás mellett, a "cimke" a "kifejezés" értékét veszi fel és felhasználható a program más részeiben, például egy adatmezőnél.

3.2.4.2. Az END direktiva

Az END direktiva használata opcionális a CP/M assemblerben, ha azonban alkalmazzuk, akkor a forrásnyelvi program utolsó utasitásaként kell szerepelnie.
A direktiva után következő összes utasitást figyelmen kivül hagyja a forditóprogram.

A direktiva a következő formákban jelenhet meg:

```
cimke     END
cimke     END     kifejezés
```

ahol a cimke alkalmazása szintén nem kötelező.

Az első formát akkor alkalmazzuk, ha azt szeretnénk,

hogy a program futtatásakor a vezérlés a ØØØØH kezdőcimre ugorjon.
A második forma esetén a "kifejezés" kiértékelése után kapott érték lesz leforditott programunk futtatásakor a kezdőcim.

Az indulási cimet tárolja az assembler forditóprogram az INTEL formátumu hexadecimális file legutolsó rekordjában.

A CP/M programok legtöbbjét az

END 1ØØH

utasitás zárja be, jelezvén a tranziens program terület kezdetét.

3.2.4.3. Az EQU direktiva

Az EQU direktivát numerikus értékek elnevezésekkel való helyettesitésére alkalmazzuk.

Általános formája a következő:

cimke EQU kifejezés

ahol a cimkének ebben a kifejezésben szabad először megjelennie, és nem lehet más utasitás cimkéje is egyben.

A "cimke", amely az esetek többségében egy név, az ember számára érthetőbbé, olvashatóbbá teszi a programot.
A direktiva egy konstans értéket rendel a cimkéhez.

3.2.4.4. A SET direktiva

A SET direktiva formája és jelentése hasonló az ORG direktiváéhoz, kivéve azt, hogy a SET utasitásban szereplő cimke a program további részében más SET utasitásokban is megjelenhet.

Általános alakja:

```
cimke      SET      kifejezés
```

A kifejezés a kiértékelése után a cimkével jelölt változó aktuális értéke lesz. Tehát amig a EQU direktivával egyetlen értéket definiálhatunk, a SET direktivával meghatározott érték addig érvényes ameddig azonos cimkenevü uj SET utasitást nem talál az assembler fordító program. Ezután a SET utasitásnál talált uj kifejezés értéket rendeli a cimkéhez.

A SET direktiva használata lehetővé teszi egyszerü változó deklarálását.

3.2.4.5. Az IF és ENDIF direktivák

Az IF és END IF direktivák a forditóprogram számára adnak információt, hogy a forditás során a forrásnyelvi program mely részeit kell figyelembe venni, illetve figyelmen kivül hagyni.

Általános alakja a következő:

```
IF kifejezés
1. utasitás
```

```
2.  utasitás
    .
    .
    .
    .
n.  utasitás
END IF
```

A forditás során a forditóprogram kiértékeli azt a "kifejezést", amely az IF direktiva után következik. A "kifejezésben" szereplő összes operandusnak a forditás korábbi fázisában értéket kell kapnia ahhoz, hogy kiszámitható legyen. Ha a "kifejezés" nem zéró értékü /igaz/, akkor az assembler a 1. 2. ... n. utasitás forditását végzi el. Ha a kifejezés értéke zéró /hamis/ akkor az 1. 2. ... n. utasitásokat listázza, de nem forditja le.
Az IF; END IF direktivapárt leggyakrabban akkor alkalmazzuk, ha "generáló programmal", amely számos paramétert figyelembe vesz, néhány paraméter beállitásával egy egyéni feladatot akarunk megoldani.

3.2.4.6. A DB direktiva

A DB direktiva lehetővé teszi a programozó számára, hogy egy vagy több byte-os adat/ok/ szöveg/ek/ számára foglaljon helyet a memóriában. A direktiva általános alakja a következő:

```
cimke    DB    e#1,   e#2, .... e#n
```

ahol e#1-től e#n-ig vagy kifejezések szerepelnek, amelyek 8 bites értéket tartalmaznak, vagy olyan ASCII karaktersorozatok, melyek hossza nem nagyobb mint 64 karakter.

A kifejezésben szereplő értékek a legutolsó program-utasitás után lesznek elhelyezve a gépi kódu programot tartalmazó file-ban, ugyanolyan sorrendben, ahogy definiáltuk őket. A karaktersorozatokat az elsőtől az utolsó karakterig tárolja a fordító program.
A két karakternél hosszabb karaktersorozatokat egyszeres apasztróf jelek közé kell zárni.

Az ASCII karaktereket ilyen tipusu definiáláskor paritás-bit nélkül helyezi el a forditóprogram és a stringekben lévő nagy és kis betüket nem alakitja át egységesen nagy betükké.

A "cimke" alkalmazása nem kötelező!

Mintapéldák szintaktikailag helyes DB direktivákra:

```
data:    DB    1, 2, 3, 4
         DB    data and ØFFh, 5,1+2
nev:     DB   'Kérlek ird le a neved'
```

Helytelen DB helyfoglalás:

```
rossz:   DB ØA245H
```

3.2.4.7. A DW direktiva

A DW direktiva hasonló a DB direktivához, de ez két byte-os /dupla pontosságu/ szavak tárolását teszi lehetővé.
Általános formája a következő:

```
cimke    DW    e#1,  e#2, .... e#n
```

ahol az e#1, ... e#n olyan kifejezéseket jelölnek, melyek 16 bites értékűek.
A DW direktivában egy-, vagy kétkarakteres karaktertipusu változókat is használhatunk. Ezeknél hosszabb karaktersorozat alkalmazását nem engedélyezi a direktiva. A két byte-os adatok tárolási formája megegyezik az I8Ø8Ø-as mikroprocesszor adattárolási formátumával: a kisebb helyiértékü byte-ot tárolja először, majd ezt követi a magasabb helyiértékü byte.

Mintapéldák:

```
dupla:    DW    dupla + 4
          DW    előjel -$
          DW    'a', 5, 'ab'
          DW    6 shl 8 or 11b
```

Helytelen DW alkalmazás

```
WRONG:    DW    'ABC'
```

3.2.4.8. A DS direktiva

A DS direktiva egy felhasználásában még nem definiált memóriaterület lefoglalását teszi lehetővé. Általános formája:

```
cimke    DS    kifejezés
```

ahol a cimke alkalmazása nem kötelező.

Az assembler forditóprogram a "kifejezésben" meghatározott mennyiségü byte kihagyásával folytatja a forditást.

A DS direktiva hatásában a következő direktiva definicióval egyetértékü:

```
cimke:    EQU $ ;  a cimke értéke
                   az aktuális cim
                   ORG $ + kifejezés
                   a foglalt hely utáni byte-ra
                   állitja a cimmutatót
```

3.2.5. Utasitás elnevezések

Az ASM program az Intel 8Ø8Ø mikroprocesszor standard mnemonik készletét fogadja el. A mnemonikok által jelölt utasitások részletes leirása az Intel cég által kiadott "8Ø8Ø Assembly Language Programming Manual"-ban találhatók.
Az egyes utasitások mnemonikjait összefoglalva itt azért irjuk le, hogy az assembler programozás szempontjából leirásunk teljes legyen.
/A következő jelölésrendszer pontos értelmezése szintén az Intel 8Ø8Ø kézikönyvben található./

Az alkalmazott jelölések a következők:

e3 egy 3 bit nagyságu értéket jelent, amely Ø és 7 között lehet és, amely az előzőleg definiált regiszterekhez A,B,C,D,E,H,L,M,SP vagy PSW-höz tartozik. Ha az utasitás regiszter-párral dolgozik, akkor a B a BC, a D a DE, a H a HL regiszterpárt jelöli.

e8 egy 8 bites értéket jelent, melynek értékhatára Ø és 255 között van

e16 egy 16 bites értéket jelent,melynek értékhatára Ø és 65535 között van.

3.2.5.1. Ugró utasitások, szubrutinhivások, visszatérő utasitások

Az ugró, szubrutinhivó és visszatérő utasitások vagy feltétel nélkül vezérlés-átadást okoznak, vagy feltétel-bitek állapota szerint megváltoztathatják a programvégrehajtás sorrendjét.

Formájuk a következő:

JMP e16 Feltétel nélküli ugrás az e16-ban meghatározott abszolut cimre.

JNZ e16 Ugrás az e16-ban meghatározott abszolut cimre, ha a zéró állapot-jelző bit értéke hamis.

JZ e16 Ugrás az e16-ban meghatározott abszolut cimre, ha a zéró állapotjelző bit értéke igaz.

JNC e16 Ugrás az e16-ban meghatározott abszolut cimre, ha a carry-bit értéke hamis.

JC e16 Ugrás az e16-ban meghatározott abszolut cimre, ha a carry bit értéke igaz.

JPO e16 Ugrás, ha az akkumulátorban lévő adat páros paritásu.

JPE el6 Ugrás, ha az akkumulátorban lévő adat páratlan paritásu.

JP el6 Ugrás az el6-ban meghatározott abszolut cimre, ha az előjel bit értéke hamis.

JM el6 Ugrás az el6-ban meghatározott abszolut cimre, ha az előjel bit értéke igaz.

A következőkben felsorolt szubrutinhivó és visszatérő utasitások feltételei megegyeznek az ugró utasitásoknál felsorolt feltételekkel.

CALL el6 Feltétel nélküli szubrutinhivás, cim el6.

CNZ el6 Szubrutinhivás, ha a zéró bit hamis.

CZ el6 Feltételes szubrutinhivás, ha a zéró bit igaz az el6 cimre.

CNC el6 Szubrutinhivás, ha a carry bit hamis értékü.

CC el6 Szubrutinhivás, ha a carry bit igaz értékü.

CPO el6 Szubrutinhivás az akkumulátor páros paritása esetén.

CPE el6 Szubrutinhivás az akkumulátor páratlan paritása esetén.

CP el6 Szubrutinhivás pozitiv eredmény esetén.

ĊM el6 Szubrutinhivás negativ eredmény esetén.

RET Feltétel nélküli visszatérés zubrutinból.

RNZ Visszatérés, ha a zéró bit hamis.

RZ Visszatérés, ha a zéró bit igaz.

RNC Visszatérés, ha a carry bit hamis.

RC Visszatérés, ha a carry bit igaz.

RPO Visszatérés, ha az eredmény páros paritásu.

RPE Visszatérés, ha az az eredmény páratlan paritásu.

RP Visszatérés, ha az eredmény pozitiv.

RM Visszatérés, ha az eredmény negativ.

RST e3 Programozott ujrainditás /restart/ utasitás, hatásában megegyezik egy CALL e3*8 utasitással.

3.2.5.2. Müveletek konstansokkal
/Immediate Instructions/

Számos olyan utasitás van, amely egyszeres vagy kétszeres pontosságu /egy vagy kétbyte-os/ regisztereket, vagy egy byte-os memóriacellákat tölt fel konstans értékekkel, illetve aritmetikai vagy logikai müveletet hajt végre egy konstans értékkel és az akkumulátorral /A regiszterrel/.

MVI e3, e8 Konstans byte-ot tölt regiszterbe /regiszter: A,B,C,D,E,H,L vagy M/.

ADI e8 Az operandust carry /átvitel/ nélkül az akkumulátorhoz adja.

ACI e8 Az operandust carry-vel adja az akkumulátorhoz.

SUI e8 Az operandust kivonja az akkumulátorból carry nélkül.

SBI e8 Az operandust kivonja az akkumulátorból carry-val.

ANI e8 Logikai ÉS müvelet az akkumulátor és az operandus byte között, az eredmény az akkumulátorban.

XRI e8 Kizáró VAGY müvelet az akkumulátor és az operandus között, az eredmény az akkumulátorban.

ORI e8 Logikai VAGY müvelet az akkumulátor és az operandus között, az eredmény az akkumulátorban.

CPI e8 Az akkumulátor összehasonlitása az operandussal /ugyanaz mint a SUI de itt nem változik meg az akkumulátor tartalma/

LXI e3, e16 Regiszterpár töltése közvetlenül a kifejezésben lévő konstans értékével /e3 egyenlő a B,D,H, SP-vel jelölt regiszterpárok valamelyikével/.

3.2.5.3. Regiszterek értékének növelése és csökkentése /Increment, decrement instructions/

Az I8Ø8Ø lehetőségeit figyelembe véve a regiszterek értékének változtatására a következő lehetőségeink vannak:

INR e3 Egyszeres pontosságu regiszter értékének növelése 1-gyel. /e3 egyike az A,B,C,D,E,H,L vagy M regiszteréknek./

DCR e3 Egyszeres pontosságu regiszter értékének csökkentése 1-gyel.

INX e3 Kétszeres pontosságu regiszter /regiszterpár/ értékének növelése 1-gyel.
/e3 egyike a B,D,H vagy SP regiszterpároknak./

DCX e3 Kétszeres pontosságu regiszter /regiszterpár/ értékének csökkentése 1-gyel.

3.2.5.4. Adatátviteli utasitások

A következő utasitások a CPU és a memória közötti adatátvitelt szolgálják.

MOV e3, e3 Adatmozgatás a baloldali operandusba a jobboldalu operandusból /e3 egyike az A,B, C,D,E,H,L vagy M regiszteréknek/.

LDAX e3 Az A regisztert tölti a regiszterpárokban meghatározott cimről.
/e3 vagy B vagy D regiszterpár./

STAX e3 Az A regiszter értékét tölti a regiszterpárban meghatározott cimre.
/e3 vagy B vagy D regiszterpár./

LHLD e16 HL regiszterpár közvetlen töltése az e16 által meghatározott cimről.
/kétszeres pontosságu töltés H és L-be./

SHLD e16 HL regiszterpár tartalmának közvetlen tárolása az e16 által meghatározott cimre.
/kétszeres pontosságu tárolás H és L-ből a memóriába./

LDA e16 "A" regiszter töltése az e16 által meghatározott cimről.

STA e16 "A" regiszter tárolása az e16 által meghatározott cimre.

POP e3 Regiszterpár töltése a stack-bõl, a Stack Pointer. változik /e3 a B,D,H vagy PSW regiszterpárok egyike/.

PUSH e3 Regiszterpár értékének tárolása a stack-ben, a Stack Pointer változik /e3 a B,D,H vagy PSW regiszterpárok egyike/.

IN e8 "A" regiszter feltöltése az e8 számu portról.

OUT e8 "A" regiszter tartalmát küldi az e8 számu porthoz.

XTHL Kicseréli az adatokat a Stack tetején a HL regiszterpár tartalmával.

PCHL A programszámláló értékét a HL regiszterpárral teszi egyenlővé.

SPHL A Stack Pointer /SP/ értékét a HL regiszterpárral teszi egyenlővé.

XCHG A DE és a HL regiszterpár tartalmát kicseréli.

3.2.5.5. Aritmetikai és logikai müveletek

A következő utasitások az akkumulátorral végeznek egyszeres pontosságu /1 byte-os/ aritmetikai, illetve logikai müveleteket.

ADD e3 Az e3 által megadott regiszter tartalmát hozzáadja az akkumulátorhoz carry nélkül /e3 egyike az A,B,C,D,E,H,L vagy M regiszte-reknek/.

ADC e3 Az e3 által megadott regiszter tartalmát hozzáadja az akkumulátorhoz carry-vel /e3 definiciója ugyanaz, mint a megelőző utasitásnál/.

SUB e3 Az akkumulátorból kivonja az e3 által megadott regiszter tartalmát carry nélkül /e3 definiciója az előzőknek megfelelő/.

SBB e3 Az akkumulátorból kivonja az e3 által megadott regiszter tartalmát carry-vel /e3 jelentése a korábbi/.

ANA e3 Az e3 által megadott regiszter és az akkumulátor közötti bitenkénti logikai ÉS müvelet /e3 jelentését lásd korábban/.

XRA e3 Az e3 által megadott regiszter és az akkumulátor bitjei közötti kizáró VAGY müvelet /e3 jelentését lásd korábban/.

ORA e3 Az e3 által megadott regiszter és az akkumulátor bytjei közötti logikai VAGY müvelet /e3 jelentését lásd korábban/.

CMP e3 Az e3 által meghatározott regiszter tartalmát és az akkumulátort hasonlitja össze és a feltétel biteket ennek megfelelően állitja be /e3 jelentését lásd korábban/.

DAA Az akkumulátor értékének decimális értékké történő átalakitása.

CMA Komplementálja az akkumulátor bitjeit.

STC A carry jelzőbitet logikai 1 értékre állitja.

CMC Komplementálja a carry jelzőbitet.

RLC Az akkumulátor bitjeit ciklikusan balra tolja a carry bit nélkül.

RRC Az akkumulátor bitjeit ciklikusan jobbra tolja a carry bit nélkül.

RAL Az akkumulátorból és a carry bitból képzett 9 bites "regisztert" ciklikusan balra tolja.

RAR Az akkumulátorból és a carry bitből képzett 9 bites "regisztert" ciklikusan jobbra tolja.

DAD e3 Kétszeres pontosságu összeadás, az e3 által kijelölt regiszter tartalmát adja a HL regiszterpár tartalmához.
/e3 lehetséges kijelölései B,D,H vagy SP/

3.2.5.6. Vezérlő utasitások

A következő vezérlő utasitások az I8Ø8Ø mikroprocesszor állapotát határozzák meg.

HLT Megállitja az I8Ø8Ø processzor futását.

DI Nem engedélyez megszakitást.

EI Engedélyezi a megszakitást.

NOP Üres utasitás, nincs müveletvégzés.

3.2.6. Hibaüzenetek

Amikor hiba van az assembly nyelvü programban, az assembler forditóprogram egybetüs hibaüzenetet küld a képernyőre. A hibás sor sorszáma is megjelenik, ez megkönnyiti a hiba analizálását.
A hibaüzenet betükódjainak jelentését a következőkben foglalhatjuk össze:

D /Data error/ adat alkalmazása nem megfelelő helyen.

E /Expression error/ egy kifejezést hibás formátumából adódóan nem lehet kiértékelni.

L /Label error/ cimke hiba, a cimke nem található a szövegben, vagy kétszer szerepel ugyanaz a cimke cimkemezőben.

N /Not implemented/ olyan lehetőségek felhasználásának igénye, amely csak későbbi ASM verziókban jelennek meg /pl. makrók/.

O /Overflow/ a kifejezés tulságosan komplikált, hogy kiszámitható legyen.

P /Phase error/ a cimke nem ugyanazt az értéket tartalmazza a program két szubszegmensében /pl. makrók, szubrutinok esetén/.

R /Register error/ fenntartott regiszternevet akarunk cimkeként használni.

V /Value error/ egy kifejezésben szintaktikailag meg nem engedett formátumu operandus szerepel.

A kezelő és a rendszer hibáira utaló hibaüzenetek:

NO SOURCE FILE PRESENT

az ASM utasitásban megadott leforditandó forrásnyelvi file nem található a lemezen.

NO DIRECTORY SPACE

a lemez directory megtelt; töröljük ki azokat a file-okat, amelyekre nincs szükség és próbálkozzunk ujra.

SOURCE FILE NAME ERROR

nem megengedett formáju .ASM file /például "?" mezővel definiáltuk/.

SOURCE FILE READ ERROR

a forrásnyelvü file nem olvasható pontosan; az assembler egy TYPE parancsot hajt végre, hogy megmutassa a hiba helyét.

CANNOT CLOSE FILE

az output file nem zárható le; vizsgáljuk meg, hogy a lemez irásvédett-e.

OUTPUT FILE WRITE ERROR

az output file nem irható a lemezre; ennek legtöbb esetben az az oka, hogy a lemez megtelt.

3.3. Interaktiv hibakereső
/Dynamic Debugging Tool - DDT/

3.3.1. Bevezetés

A DDT program CP/M környezetben biztositja a programok könnyü ellenőrzését és hibajavitását. A programot a következő utasitások egyikével aktivizálhatjuk:

```
DDT
DDT file-név.HEX
DDT file-név.COM
```

ahol a filenév a program neve, amelyet vizsgálunk. A DDT program minden esetben - standard memóriakiosztás esetén - a Console Command Processor /CCP/ helyére kerül. Igy helyet szabadit fel és közvetlenül használja a Basic Disc Operating System /BDOS/ szolgáltatásait.

A betöltés után a DDT program a következő üzenettel jelentkezik be:

```
nnK   DDT-s      VER m.m
```

ahol "nn" a memória mérete kbyte-ban
"s" a hardware rendszerre utaló mutató; lehetséges értékei:

D - Digital Research standard verzió
M - MDS verzió
I - IMSAI standard verzió

O - Omron rendszer
S - Digital System standard verzió

az "m.m" a DDT verziószáma.

A bejelentkező üzenet után a DDT egy "-" karakterrel jelzi, hogy készen áll a különböző parancsok végrehajtására.
Az operátor egykarakteres utasitásokat adhat meg, amelyeket a DDT program a CR-billentyü lenyomása után azonnal végrehajt.

A DDT-vel való kommunikáció közben alkalmazhatók a CP/M vezérlő /control/ karakterei is:

RUBOUT	törli a legutoljára leütött karaktert
CTRL-U	törli az aktuális sort
CTRL-C	ismételten betölti a rendszert.

Bármely utasitássor maximum 32 karakter hosszuságu lehet /egy automatikus kocsi-vissza soremelés karaktert generál a rendszer 33. karakterként/, ahol az első karakter a következő utasitások egyike:

A	Assembler mnemonikokkal és operandusokkal történő programozásra nyujt lehetőséget.
D	A memória tartalmát irja ki hexadecimális és karakteres formában.
F	A memóriaterületet konstans adattal tölti fel.
G	Az utasitások végrehajtásának meginditása opcionális töréspontta l.

I	File-név beolvasás az operátori konzolról.
L	Kilistázza a memóriát assembler mnemonikok formájában.
M	Programszegmenst mozgat adott memóriaterületről adott memóriaterületre.
R	Adott file beolvasása lemezről /a név az "I" parancsban lett megadva/.
S	A memóriaterület felülirása uj értékekkel.
T	A program futásának lépésenkénti követése a regiszterek megjelenitésével /trace/.
U	A program futásának lépésenkénti követése a regiszterek megjelenitése nélkül /untrace/.
X	A CPU állapotának megjelenitése és opcionális változtatása.

A fenti parancsok paraméterei szóközzel vagy vesszővel elválasztott hexadecimális számok lehetnek, ezeket a részletes leirásnál fogjuk ismertetni.

A DDT program numerikus eredményei hexadecimális formátumuak. Az utasitást csak akkor hajtja végre a program, ha az utasitássor végére kocsi vissza soremelést gépeltünk.
A program nyomkövetésének bármely pontján az operátor megállithatja a DDT program müködését a CTRL-C vagy a GØ parancsok egyikével. /GØ = ugrás a ØØØØH cimre/. Ebben az esetben az operátor lemezre mentheti a memória tartalmát a SAVE paranccsal.

A SAVE parancsban lévő file nevében kiterjesztésként ".COM" nevet alkalmazzuk, igy a memóriatartalom adott cimre /1ØØH/ való visszatöltése egyszerüen a filenév leirásával történhet meg. Ha a memóriatartalom ujabb vizsgálata szükséges, akkor a

DDT file-név.COM

paranccsal tölthetjük ujra a memóriába az elmentett "memóriaképet".

3.3.2. DDT parancsok

A DDT programban a parancsokat a megelőző parancs végrehajtását visszajelző karakter /"-"/ megjelenése után lehet megadni.
A parancsok többsége a "CPU állapot" elnevezésü memóriaterület tartalmát változtatja meg. Ezek a memóriaelemek modellezik a központi egység belső regisztereit. A program indulásakor minden "CPU állapot" elemet nulláz, kivétel ez alól a PC /Program Counter/, illetve az SP /Stack Pointer/, ezek értékeit nem változtatja. A PC regiszter tartalma az éppen betöltött program indulási cime, amelyet a betöltött hexadecimális file /.HEX/ utolsó rekordja tartalmaz /lásd bővebben az I és az R parancsoknál./

3.3.2.1. Az A /Assemble/ parancs

A DDT programban az"A" parancs egysoros assemblálásra nyujt lehetőséget. A parancs formája a következő

As

ahol s az a hexadecimális indulási cim, ahonnan az assemblálást el akarjuk kezdeni. A program visszairja a képernyőre a cimet, és egy assembler mnemonikra, valamint egy hexadecimális formátumu operandusra vár.

Ezek begépelése után leforditja a mnemonikot, megkeresvén annak hexadecimális kódját és a következő üres memóriacim kiirásával ujabb parancsra vár.

Az "A" parancs állapotból üres sor leirásával léphetünk ki a parancsválasztási állapotba.

Az assembler forditás helyességéről az operátor az "L" parancs segitségével győződhet meg, amely mnemonikok és hexadecimális operandusok formájában listázza visszza a memória tartalmát.

3.3.2.2. A D /Display/ parancs

A "D" parancs lehetővé teszi az operátor számára, hogy a memóriaterület egy részét hexadecimális és karakteres formában megjelenitse. A parancs formái a következők:

D

Ds

Ds,f

Az első esetben a memóriát az aktuális PC értéktől /CPU regiszterek PC paramétere - X paranccsal

kiiratható/ ez a DDT legelső betöltésekor lØØH, irja ki a program, amelyet 16 kiirható sor követ.
A megjelenitett sorok formátuma a következő:

```
          16 byte                          16 karakter
aaaa      bb bb ... bb                     cc ... c
```

ahol "aaaa" az a memóriacim hexadecimális formában, ahonnan kezdve a program 16 byte-ot kiir /1 byte-nak két hexa számjegy - bb - felel meg/ a "c" betüvel jelölt karakterek a hexadecimális formában kiirt memóriatartalom karakteres megfelelői. A grafikus formában nem ábrázolható karakterek /például tabulátor, vezérlőkarakterek/ helyére "." karaktert nyomtat a program.

A kisbetük helyett is nagybetüket ir a program a karakteres megjelenités mezejében.

A "D" parancs második formája teljes egészében megegyezik az elsővel, de itt a megjelenités kezdőcime "s" lesz.

A harmadik forma alkalmazása esetén az "s" cimtől az "f" cimig terjedő tartalom jelenik meg.

3.3.2.3. Az F /Fill/ parancs

Az "F" parancs formája a következő:

```
Fs,f,c
```

ahol "s" a kezdeti cim, "f" a végső cim és "c" a hexadecimális konstans, amellyel feltöltjük e cimek közötti memóriaterületet.

3.3.2.4. A G /Go/ parancs

A memóriában lévő program futását a "G" paranccsal indithatjuk el. A parancs formái a következők:

G
Gs
Gs,b
Gs,b,c
G,b
G,b,c

Az első forma a programszámláló aktuális értékétől és az aktuális CPU állapotból /regiszter tartalmakról/ inditja el a program futását töréspont nélkül.

A második forma tartalmilag hasonló az elsőhöz, de a program számláló /PC - Program Counter/ kezdőértéke "s" lesz. A harmadik forma alkalmazása esetén a program futása az "s" cimen kezdődik és a "b" cim elérése után a futás megszakad és a vezérlést visszaadja a DDT programnak.

A negyedik forma alkalmazásával két töréspont beiktatására nyilik lehetőségünk, az elsőt a "b", a másodikat pedig a "c" cimre.
Az utolsó két parancsformátumban a PC aktuális értékét használja fel indulási állapotként a "G" parancs, és egy vagy két töréspont beiktatására nyilik lehetőség.

A program végrehajtása az indulási cimtől valós időben történik a következő töréspontig.

Ha a program futása során nem érint egyetlen töréspontot sem, akkor nem tud visszatérni a DDT programhoz. Ebben az esetben egy hardware RST 7 utasitás /gépi kódja ØFFH/ segitségével lehet a futást megállitanunk /kell lennie a gépen egy olyan billentyünek, amelynek ez a hatása/. Ha viszont töréspontot ér el a program, akkor

*d

állapottal jelentkezik az operátori konzolon, ahol "d" a cim, ahol abbahagyta a futást.
Az operátor megvizsgálhatja a CPU állapotregisztereit, igy információt kaphat a program állapotáról /X parancs/.

3.3.2.5. Az I /Input/ parancs

Az "I" parancsban az operátor egy filenevet adhat meg, amelyet a későbbiekben a beolvasási parancs használ fel /az "R" parancs/.
Az "I" parancs formái a következők:

Ifilenév
Ifilenév.kiterjesztés

Ha a második formát használjuk és a file kiterjesztése vagy ".HEX" vagy ".COM" akkor az "R" parancs használatakor egyszerü bináris vagy hexadecimális adatokat olvasunk.

3.3.2.6. Az L /List/ parancs

Az "L" parancsot a memóriatartalom assembly mnemonikok formájában való listázására alkalmazzuk. A parancs formái a következők:

L
Ls
Ls,f

Az első parancsforma 12 assembler sort "fordit vissza" a gépi kódu utasitásokból, az aktuális PC értékétől kezdődően. A második forma a PC értékét egyenlővé teszi az "s"-sel és innen kezdve ir ki 12 sort.
A legutolsó forma alkalmazása esetén a listázás az "s" cimtől az "f" cimig történik.

Mindhárom esetben a további listázáshoz egy ujabb "L" parancsot kell adni.

3.3.2.7. Az M /Move/ parancs

Az "M" parancs lehetővé teszi, hogy egy memóriablokkot, amely program- vagy adatblokk, a számitógép operativ memóriájában egy másik cimre helyezzünk át. A parancs formája a következő:

Ms,f,d

ahol "s" a mozgatásra kerülő blokk kezdőcime, "f" a végcime, "d" pedig az a kezdőcim, ahová a blokkot mozgatni kell.

Az első adatot az "s" cimről a "d" cimre mozgatja a program, majd a cim növelésével a többi byte-ot is sorban lemásolja az uj memóriaterületre.

3.3.2.8. Az R /Read/ parancs

Az "R" parancs értelmesen az "I" paranccsal együtt használható és a lemezegységről olvashatunk be segitségével ".COM" vagy ".HEX" kiterjesztésü file-okat. A parancs formái a következők:

R
Rb

ahol a "b" paraméter alkalmazása opcionális, ez egy ugynevezett báziscim, amelyet minden egyes program vagy adatcimhez hozzáad a program, amely beolvasásra került. A beolvasási utasitás nem irhatja felül a memóriaterület első lapját /ØØØØH - ØØFFH közötti területet/.

Az "R" parancs végrehajtásához feltétlenül szükséges egy "I" parancs előzetes végrehajtása, ahol meghatározzuk a file-nevet és kiterjesztést.

Az "R" parancs feltételezi, hogy ha a file kiterjesztése nem ".COM", akkor a file Intel hexadecimális formátumu, a ".COM" file pedig egyszerü bináris adatsorozat.

Az "R" parancsot közvetlenül a DDT program hivásá is aktivizálhatjuk a következő formában:

DDT file-név.kiterjesztés

amely tökéletesen ekvivalens a következő utasitás-sorozattal

```
DDT
- Ifile-név.kiterjesztés
- R
```

A file sikeres beolvasása után egy üzenetet küld a DDT program, melynek formája:

```
NEXT PC
nnnn pppp
```

ahol nnnn a betöltött file-t követő első szabad cim, a pppp pedig a PC értékként beállitott cim. Ez ".COM" file-oknál általában 1ØØH, egyéb esetben a ".HEX" file utolsó rekordjában van tárolva. Sikertelen file beolvasás esetén "?" karakterrel jelzi a program a hibát.

3.3.2.9. Az S /Set/ parancs

Az "S" parancs lehetővé teszi, hogy megvizsgáljuk és megváltoztassuk a memória egy adott cimü byte-ját. A parancs formátuma a következő:

```
Ss
```

ahol "s" a hexadecimális értékü indulási cim. Ha az operátor CR-t üt be, akkor az adott cimen lévő

tartalom változatlan marad. Ha egy byte-ban tárolható hexadecimális számot gépel, akkor ez a megadott cimen közvetlenül tárolódik.

Mindkét esetben a DDT program a következő cim tartalmának felajánlásával folytatja müködését. A DDT program parancsválasztási állapotához való visszatérés egy pont /"."/ karakter leütésével lehetséges, vagy abban az esetben, ha nem értékelhető bemeneti adatot adtunk meg /pl. 3 hexadecimális számjegyet a kettő helyett/.

3.3.2.10. A T /Trace/ parancs

A "T" parancs lehetővé teszi, hogy egy program lépésenkénti végrehajtásánál a CPU állapotregiszterek tartalmát, változását figyelemmel kisérhessük. Parancs formái a következők lehetnek:

```
T
Tn
```

ahol az első esetben a CPU állapotot jeleniti meg a program és a következő programlépést végrehajtja.

A programlépés végrehajtása után a parancs visszaadja a vezérlést a DDT programnak, amely a

```
* kkkk
```

formáju kiirással jelentkezik vissza, ahol "kkkk" a következő végrehajtható utasitás cime.

A "T" parancs második formája hasonló az elsőhöz, azzal a különbséggel, hogy ebben az esetben "n" darab lépés végrehajtása történik meg /"n" egy hexadecimális érték/.

A trace módból való kilépésre a RUBOUT billentyü használható.

A CPU állapot megjelenitése mindig a programlépés végrehajtása előtt történik meg. A megjelenitès formátumát a későbbiekben az "X" parancsban mutatjuk be.

A "T" parancs alatt futó program kb. 5ØØ-szor lassabban fut, mint a valós idejü futás, mivel a DDT program minden programlépés végrehajtása után különféle ellenőrzéseket, memóriatöltéseket végez.

Megszakitást kezelő szubrutinok vizsgálata is lehetséges, amennyiben azok nem változtatják meg a CP/M rendszer, illetve a DDT program müködtetéséhez szükséges feltételeket.

További problémát jelenthet, hogy programkövetési módban a megszakitás engedélyezett, igy a program müködését zavarhatják az aszinkron módon jelentkező megszakitás kérések.

Abban az esetben, ha az operátor a RUBOUT billentyüt használja, vagy "RST 7" utasitás végrehajtást kezdeményezi hardware uton, a trace müvelet aktuális utasitásának végrehajtását még befejezi a DDT program.

3.3.2.11. Az U /Untrace/ parancs

Az "U" parancs, akár a "T", szintén lépésenkénti programvégrehajtást tesz lehetővé, de nem jeleniti meg a lépések után a CPU állapotregiszterek tartalmát.

Az "U" parancsmódban 1-től 65535-ig /ØFFFFH/ terjedő lépésszámu programlépés szimulálására nyilik lehetőség. Minden, a korábbiakban a "T" parancsnál ismertetett feltétel az "U" parancsra is érvényes.

3.3.2.12. Az X /Examine/ parancs

Az "X" parancs lehetővé teszi, hogy megjelenitsük, illetve megváltoztassuk a CPU állapotregiszterek tartalmát. A parancs formái a következők:

X
Xr

ahol "r" az I8Ø8Ø CPU regiszterek egyike.

C	Carry bit	Ø/1
Z	Zero bit	Ø/1
M	Minus bit	Ø/1
E	Paritás bit	Ø/1
I	Decimális carry bit	Ø/1
A	Akkumulátor	Ø - FF
B	BC regiszterpár	Ø - FFFF
D	DE regiszterpár	Ø - FFFF
H	HL regiszterpár	Ø - FFFF
S	Stack mutató	Ø - FFFF
P	Program számláló	Ø - FFFF

Az első esetben a CPU regiszterek állapota a következő formában jelenik meg

CfZfMfEfIf A=bb B=dddd D=dddd H=dddd S=dddd P=dddd utasitás

ahol "f" egy bitet jelöl, melynek értéke Ø vagy 1, "bb" egy byte érték és "dddd" egy két byte-os mennyiség, amely a különböző regiszterpárokhoz tartozik. Az "utasitás" mező tartalmazza azon utasitás mnemonikus formáját, amelyre a CPU állapotregiszterek közül a program számláló mutat.

A "X" parancs második formájával az "r"-rel megadott CPU állapotregisztert lehet kiiratni vagy megváltoztatni.

Az utasitásban az "r" jelentése a C,Z,M,E,I,A,B,D, H,S vagy P karakterek egyike.

A regiszterek mindegyikénél először megjelenik a regiszter aktuális állapota, majd ezután a DDT program a klaviaturáról bemeneti adatot fogad el. Ha a változtatásként megadott érték nem esik az egyes regiszteréknél megadott értékhatárok közé, akkor a módositandó regiszter állapota nem változik meg /korábbi értékét megtartja/.

3.4. A hexadecimális programokat betöltő LOAD program

Az "ASM" program által előállitott ".HEX" kiterjesztésü file-t az operativ memóriába /TPA/ a "LOAD" program

segitségével tölthetjük be. A "LOAD" program a hexadecimális byte-okat tartalmazó ".HEX" file-t alakitja át közvetlenül futtatható ".COM" file-lá.
A folyamatot a 3.4.1. ábra szemlélteti

3.4.1. ábra

A "LOAD" program, amely maga is egy tranziens program /LOAD.COM/ a következő módon használható:

A "LOAD" programnév legépelése után a betöltendő program nevét kell leirnunk. A "LOAD" program feltételezi, hogy a betöltendő program ".HEX" kiterjesztésü.

Egy betöltési parancs igy például:

A > LOAD PROG ↵

A "LOAD" program ezen utasitássorra megkeresi a kurrens lemezegységen a "PROG.HEX" nevü file-t, és létrehoz egy ujabb file-t "PROG.COM" néven. A "PROG.COM" file az un. közvetlenül betölthető és futtatható kódokat tartalmazza. A "PROG.COM" file segitségével a "PROG" nevü program a file-név begépelésével betöltődik, és futása elindul.

Az adott filenévhez ".HEX" kiterjesztésü file-ból ".COM" kiterjesztésü file előállitásához csak egyszer kell a LOAD programot alkalmazni.

A ".HEX" file-nak Intel hexadecimális formában kell az adatokat tartalmaznia. Ilyen file-t készit az ASM program.

A ".HEX" file-t másik egységről is betöltheti a "LOAD" program, ebben az esetben a file-névnek tartalmaznia kell az illető lemezegység azonositóját is.
Például:

A LOAD B:GAMES

Ez az utasitás a "B" azonositóju egységről betölti a "GAMES.HEX" nevü file-t és a "A" egységen /ez a kurrens egység/ létrehoz egy "GAMES.COM" file-t.

Annak a gépi kódu programnak, amelyet a "LOAD" program segitségével "átalakitunk", rendelkeznie kell olyan programrekorddal, amely a 1ØØH cimre tölt információt. Ha nincs ilyen rekord, akkor a "LOAD" program a 1ØØH cimtől az első programlépésig zérusokkal tölti fel a memóriát /NOP = nincs müveletvégzés utasitással/.
Igy biztositja azt a futási feltételt, amely a ".COM" file-oknál szükséges, nevezetesen azt, hogy a file betöltése után "1ØØH" cimről induljon a program futása.

3.5. A memória listázását biztositó DUMP program

A "DUMP" program kilistázza az operátori display-re

egy file tartalmát hexadecimális formában. Tetszőleges file listázható. A képernyőn egy sorba 16 byte kerül, a sor bal szélén az első byte abszolut cime jelenik meg.

A "DUMP" program használatára nézzük a következő példát:

A DUMP B:PROG.HEX

A "DUMP" program a "B" azonositóju egységben lévő lemezről beolvassa a "PROG.HEX" nevü file-t és a "DDT" program "D" paraméternél bemutatott formátum szerint kilistázza az operátori display-n.

IV. R É S Z :

A CP/M PARANCSOK ÉS RENDSZER-PROGRAMOK ÖSSZEFOGLALÁSA

Ebben a részben az előzőekben bemutatott CP/M parancsokat és rendszerprogramokat foglaljuk össze ABC sorrendben.

Az egyes parancsok mellett fel van tüntetve, hogy melyik verzióban vannak meg /1.4 vagy 2.2/. A parancsok és rendszerprogramok funkcióleirását követi azok kiterjesztése: pl. rezidens parancs vagy .COM file.

Az argumentumok felsorolásakor az opcionálisakat /elhagyhatókat/ szaggatott vonallal történő aláhuzás jelzi. A kapcsos zárójelbe tett argumentumlista azt jelzi, hogy a felsorolt argumentumok közül legalább egynek szerepelnie kell /ha nem mind opcionális/.

A felhasználó által begépelendő üzeneteket folyamatos vonallal történő aláhuzás jelzi. A CR/Carriage Return - Kocsi vissza/ karaktert ↵ jelöli, a Control billentyü és egy másik billentyü egyidejü lenyomását pedig CTRL pl. CTRL-C.

Ha az argumentumok között file-név szerepel, az mindig tartalmazhat egy, a lemezegységre utaló betüt /pl. B:file/, amely egy, a kurrenstől eltérő lemezegységet definiál, kivéve, ha az argumentum kifejezetten tiltja alternativ lemezegység megadását.

A .COM file-okban tárolt programokat akkor lehet végrehajtani, ha rajta vannak a kurrens, vagy egy másik egységben levő lemezen. A kurrens lemezegység az az egység, amelynek betüjele megjelenik a promptban, pl. B esetén ez a B egység. Megváltoztatása egyszerüen ugy hajtható végre, hogy begépeljük az uj lemezegységet jelölő

betüt, utána egy kettőspontot és egy CR-t, pl.:

A> C:↵

hatására

C >

prompt jelenik meg és a C lesz a kurrens lemezegység.

Pl., ha a MINTA.COM file-ban lévő programot szeretnénk végrehajtani, elegendő a MINTA↵ üzenet begépelése. Ha a MINTA.COM nevü file a B lemezen van és az A lemezegység a kurrens, akkor B:MINTA↵ üzenet hatására történik meg a MINTA nevü program futtatása.

1. ASM CP/M 1.4
CP/M 2.2

Egy file assemblálása

/ASM.COM/

Formátum:

1. ASM file-név
2. ASM file-név .shp

Argumentumok:

file-név Egy .ASM kiterjesztésü forrásfile /szövegfile/ neve, amely assembly nyelvü utasitásokat tartalmaz ASCII kódolásu szövegként. A .ASM kiterjesztést nem kell megadni a file-névben.

.shp

Az ASM opcionális paraméterei: egy pont és utána három betü. Az s helyén szereplő betü /A,B,.../ annak a lemezegységnek a jelölésére szolgál, amely a forrás /.ASM/ file-t tartalmazza, ha ez a file nem a kurrens lemezegységen van. A h helyén szereplő betü /A,B,.../ azt a lemezegységet definiálja, amelyre az eredményül keletkezett .HEX kiterjesztésü file kerül. Ha ezen a helyen Z szerepel, az arra utasitja az ASM-et, hogy ne generáljon .HEX file-t. A p helyén álló betü /A,B,.../ azt a lemezegységet jelöli, amelyre az ASM által készitett .PRN kiterjesztésü file kerül. Ha itt X szerepel, akkor az ASM a terminálra /display-re/ küldi a .PRN file-t. Z esetén az ASM nem generál .PRN file-t.

Leirás:

Az assembler program /ASM.COM/ egy Intel-8Ø8Ø vagy Z-8Ø assembly nyelven irt programot tartalmazó forrás file-ból egy .HEX kiterjesztésü, gépi kódot tartalmazó file-t hoz létre. Ezt a .HEX file-t később a LOAD parancs segitségével be lehet tölteni a rendszerbe tranziens parancsként /végrehajtható programként/. Az ASM egy .PRN kiterjesztésü lista file-t is készit, amely az assembly nyelven irt forrássorokat és a nekik megfelelő gépi kódot tartalmazza. Az assembley szintaktikának nem megfelelő sorok meg vannak jelölve ebben a file-ban /hibajelzés/.

Ha a .ASM kiterjesztésü forrás file a kurrens lemezen

van, és létre akarjuk hozni a .HEX és a .PRN file-t is, az 1. parancsformátumot kell használnunk. Ha az előbbi feltétel nem teljesül, a 2. parancsformátumot kell használnunk, és meg kell adnunk a forrás file-t, a .HEX és a .PRN file-t tartalmazó egységet az ott ismertetett módon. Ha csak .PRN file-t kívánunk létrehozni /tehát .HEX file-t nem/, akkor a h helyén Z-t kell megadnunk. Ha viszont csak .HEX file-t akarunk generálni, akkor p helyén Z-t kell szerepeltetnünk. Ha a .PRN kiterjesztésü file-t a display-re akarjuk küldeni /képernyőre listázás/, akkor p helyén X-et kell megadnunk. Ekkor a file nem őrződik meg lemezen. Mindkét formáju parancsmegadás esetén az ASM a forrásprogram sorait az Intel hexa formátumnak megfelelően forditja gépi kódra. Ha hibás sort talál a forrás file-ban, kiirja a hibás sort és a hiba kódját.

Példák:

1./ A > ASM PROG ↵

hatására leforditódik a kurrens lemezen levő PROG.ASM nevü file.

2./ A > ASM PROG.ABZ ↵

hatására leforditódik az A lemezegységen levő PROG.ASM nevü file, az eredményül keletkezett PROG.HEX file a B lemezre kerül, PROG.PRN nevü file pedig nem keletkezik.

2. DDT

CP/M 1.4.
CP/M 2.2

Hibakereső /debugger/ program programok betöltésére, megváltoztatására és tesztelésére
/DDT.COM/

Formátum:

DDT file-név

Argumentum:

file-név — Opcionális argumentum egy .COM vagy .HEX kiterjesztésü file definiálására. A kiterjesztést is meg kell adni.

Leirás:

A DDT a CCP helyére kerül a memóriában, és betölti a megadott file-t a TPA-ba. Ha nincs megadva a file-név, a DDT vár egy file megadására, amelyet be kell töltenie a memóriába. A DDT-nek saját belső parancsai vannak a memóriahelyek tartalmának megjelenitésére és megváltoztatására, töréspontok beiktatására és más hibakereső funkciók végrehajtására.

A DDT végrehajtásához annak programként léteznie kell valamelyik elérhető lemezen. A DDT terminálására /a hibakeresés befejeztekor/ a GØ parancsot kell begépelni.

Példa:

A > DDT PIP.COM ↵

Megjegyzés: további információk a DDT-ről a kézikönyv assembly nyelvű programozásról szóló fejezetében találhatók.

3. DIR

CP/M 1.4
CP/M 2.2

A kurrens lemezegység directoryjában levő file-nevek kilistázása a display-re
/Rezidens parancs/

Formátum:

DIR { file-név | csoportos file-hivatkozás }

Argumentumok:

file-név — Opcionális argumentum, melynek hatására a DIR csak a file-névvel megnevezett file-t irja ki.

csoportos file-hivatkozás — A file-név opcionális helyettesitése, melynek hatására a DIR számos file nevét listázza ki. Mind a file-név mind a csoportos file-hivatkozás tartalmazhat lemezegység-azonositót.

Leirás:

Ha file-név vagy csoportos file-hivatkozás nincs megadva, a DIR az összes olyan file nevét kiirja, amely a kurrens lemezegység directory-jában van /csak a $DIR attributumu file-okat, a $SYS attributumuakat nem/. Megjegyezzük, hogy a 2.2 verzió csak a kurrens

felhasználói terület file-jait irja ki.

Ha file-név is meg van adva, a DIR csak ezt a file-t irja ki, ha az a kurrens lemezen és a kurrens felhasználói területen van /a 2.2 verzióban/. Ha lemezegység-azonositót is megadtunk /A:,B:,.../ a DIR ezen a lemezen, a kurrens felhasználói területen keres.

Ha file-név helyett csoportos file-hivatkozást adtunk meg, a DIR az összes olyan file-t felsorolja, amely benne van a csoportban, és a kurrens /vagy a specifikált/ lemezegységen, a kurrens felhasználói területen van.

A CP/M 1.4 és 2.2 verziójában egyaránt a DIR beépitett parancs /azaz része az operációs rendszernek/, és bármely lemezről vagy felhasználói területről végrehajtható.

Megjegyzés: Az 1.4 verzióban a DIR egy sorba csak egy file-nevet ir, mig a 2.2 verzióban egynél többet.

Példák:

```
A > DIR↵
ASM.COM     DUMP.COM     ED.COM      PIP.COM
LOAD.COM    PROG.HEX     FILE.TXT    GAME.BAS
.
.
.
```

/A CP/M 2.2-es verziója a fenti formátumban irja ki az összes $DIR attributumu, az A lemezegység Ø-s felhasználói területén lévő file-t./

```
B > DIR *.TXT↵
SAMPLE.TXT
PROG.TXT
DOCU.TXT
.
.
.
```

/A CP/M 1.4-es verziója a fenti formátumban irja ki a B lemezegységen levő összes .TXT kiterjesztésü file-t./

```
A > DIR B:↵
B:  ASM.COM
B:  PIP.COM
B:  DOCU.TXT
A >
```

4. DUMP

DP/M 1.4
CP/M 2.2

Egy lemezen levő file kilistázása a terminálra hexadecimális formában
/DUMP.COM/

Formátum:

DUMP file-név

Argumentum:

file-név Tetszőleges, lemezen levő file neve /a kiterjesztéssel együtt/.

Leirás:

A DUMP parancs tetszőleges, lemezen levő file tartalmának megjelenitésére szolgál. A file display-re listázódik, hexadecimális formában. Egy sorba 16 byte kerül, a bal oldalon az első byte abszolut cime látható.

A DUMP programot parancsként /DUMP.COM/ kell végrehajtani, megadva a file nevét,kiterjesztésével együtt. Ha a DUMP.COM file nincs a kurrens lemezen, akkor a parancsot megelőzően meg kell adni a lemezegység azonositóját is.

Példák:

A > DUMP PROG.HEX↵

A > DUMP B:SAMPLE.COM↵

5. ED

CP/M 1.4
CP/M 2.2

Egy file szerkesztése /editálás/
/ED.COM/

Formátum:

ED file-név

Argumentum:

file-név: A szerkesztendő file neve /szövegfile-nak vagy más ASCII-file-nak kell lennie/. Meg kell adni a kiterjesztést is.

Leirás:

Az ED program egy szövegpuffert hoz létre, a felhasználó módosithatja a pufferban levő szöveget. Az ED először törli az összes .BAK kiterjesztésü file-t, amelynek neve /kiterjesztés nélkül/ azonos a szerkesztett file nevével /pl. ha a szerkesztett program neve MINTA.TXT, akkor a MINTA.BAK nevü file törlődik/. Ezután lehetséges a pufferben levő szöveg módositása; könyvtári file-okból is beszerkeszthetünk részeket. Ha az E paranccsal kilépünk az editorból, az ED lezárja az uj forrás file-t és létrehoz egy "backup" file-t az eredeti forrás file megőrzésére.

Az ED.COM programnak egy hozzáférhető lemezen kell lennie. Az ED használatát részletesen a III. részben ismertettük.

6. ERA

CP/M 1.4
CP/M 2.2

Egy vagy több file törlése a lemezről
/Rezidens parancs/

Formátum:

ERA { file-név | csoportos file-hivatkozás }

Argumentumok:

file-név — A törölni kivánt file neve. Tartalmaznia kell a kiterjesztést is, meg lehet adni egy lemezegység-azonositót is a parancsban.

csoportos file-hivatkozás — Egynél többfile törlésekor használható. A *.* megadás esetén az 1.4 verzióban az összes, a kurrens lemezen levő file törlődik; a 2.2-ben csak a kurrens felhasználói terület file-jai törlődnek, a többi felhasználóé nem.

Leirás:

Az ERA parancs törli a megadott file-t, ha az nem "csak olvasható" /read only, azaz $R/O attributumu/, vagy a specifikált lemez nem "csak olvasható". Ha az ERA nem találja meg a törlésre kijelölt file-t, a következő üzenetet irja ki: "No File". A 2.2 verzióban az ERA parancs csak a kurrens felhasználói területen levő file-okat képes törölni. Alternativ egységeken levő file-ok is törölhetők, ha a lemezegység-azonositót megadjuk a file-név részeként. /Pl. az ERA B:MINTA.* parancs törli az összes MINTA nevü file-t - bármilyen kiterjesztése is van - a B lemezen levő kurrens felhasználói területen/.

Az ERA beépitett parancs mind az 1.4 mind a 2.2 verzióban, igy bármelyik lemezegységről végrehajtható.
Az egész lemez törlése az 1.4 verzióban egyszerüen a *.* file-megadás segitségével érhető el. Ugyanez a 2.2-ben ugy lehetséges, hogy irunk egy programot, amely feltölti a lemezt valamilyen adattal. Másik lehetőség ebben a verzióban az ERA *.* parancs megadása az összes felhasználói területen, ha biztosak vagyunk abban, hogy nincs "csak olvasható" /$R/O attributumu/ file.

Példák:

A > ERA MINTA.TXT↵

/Törli az A lemezen levő MINTA.TXT file-t./

A > ERA B:PROG.TXT↵

/Törli a B lemezen levő PROG.TXT nevü file-t./

A > ERA *.HEX↵

/Törli az összes .HEX kiterjesztésü file-t az A lemezegységen./

7. LOAD

CP/M 1.4
CP/M 2.2

Egy file betöltése a memóriába, .HEX file konvertálása .COM /végrehajtható parancs/ file-lá /LOAD.COM/

Formátum:

LOAD file-név

Argumentum:

file-név — .HEX kiterjesztésü file neve. A kiterjesztést nem kell megadni.

Leirás:

A LOAD program egy Intel hexadecimális formátumban levő /.HEX kiterjesztésü/ file-t parancs file-lá /.COM kiterjesztésü file/ konvertál.

A LOAD parancs végrehajtható alternativ lemezegységről is, ha lemezegység-azonositót is megadunk. A LOAD által generált .COM file-t tranziens parancsként használhatjuk, végrehajtása egyszerüen a file-név begépelésével indítható.

Példa:

A > LOAD MINTA ↵

/MINTA.HEX konvertálása MINTA.COM-má./

A > MINTA ↵

/A MINTA nevü program elinditása./

8. MOVCPM

CP/M 1.4
CP/M 2.2

CP/M adott verziójának rekonfigurálása memóriamegyság változása esetén
/MOVCPM.COM/

Formátum:

MOVCPM { * / bb } *

Argumentumok:

bb — Opcionális memóriaméret-megadás decimálisan /pl. 32, ha 32K memória van/. Értéke 16 és 64 között kell legyen. Ha helyettes * szerepel, a MOVCPM meghatározza a host mikrogép RAM területének nagyságát, és erre épiti föl a CP/M-et.

x A második csillag - amely akár a bb, akár az első csillag argumentum után szerepelhet - arra utasitja a MOVCPM-et, hogy hagyja az ujonnan generált rendszert a memóriában egy SYSGEN vagy egy SAVE utasitás végrehajthatósága érdekében. Ez az argumentum is opcionális. Ha elhagyjuk, a MOVCPM nem irja föl az uj rendszert a lemezre.

Leirása:

A MOVCPM program létrehozza a rendszer memóriaképét, majd rekonfigurálja azt annak érdekében, hogy illeszkedjék az uj RAM mérethez, amely vagy a bb argumentum, vagy a host mikrogép maximális RAM területe.

A MOVCPM.COM végrehajtható, ha rajta van egy hozzáférhető lemezen. Leggyakrabban akkor használjuk, ha egy másik hardware konfigurációra akarjuk generálni a CP/M-et.

Példák:

A > MOVCPM48 ↵

/Létrehozza a CP/M 48K-s verzióját, majd elinditja azt anélkül, hogy fölirná a lemezre./

A > MOVCPM32 x ↵

/32k-s CP/M-et generál, és azt a memóriában hagyja egy későbbi SYSGEN vagy SAVE müvelet végrehajtása céljából./
A végrehajtás után a következő üzenet jelenik meg:
READY FOR "SYSGEN" OR
"SAVE32 CPM32.COM"

A > MOVCPM xx ↵

/Maximális memóriaméretet feltételező /CP/M-et generál, és memóriában hagyja azt a későbbi SYSGEN vagy SAVE céljára./

Megjegyzés: További információk a CP/M installálása és módositása cimü fejezetben találhatók.

9. PIP

CP/M 1.4
CP/M 2.2

Perifériakezelő program
/PIP.COM/

Formátum:

1./ PIP { d: newcopy / d: } = d:oldcopy [p] ↵

2./ PIP ↵

* d: newcopy=d: oldcopy [p] ↵

* ... ↵

* ... ↵

* ↵

3./ PIP { dev: / d:file-név } = { dev: / d:file-név } [p], { dev: / d:file-név } ,... [p] ↵

4./ PIP d:=d: csoportos-hivatkozás [p] ↵

Argumentumok:

{d:newcopy / d:} Az 1. és 2. formátum alkalmazása esetén választani lehet, hogy a másolatnak uj neve legyen-e /newcopy/, vagy azonos néven, de egy másik lemezre /a d: lemezre/ kerüljön-e.

d: oldcopy Az 1. és 2. formátum esetén a másolandó file nevét meg kell adni, a lemezegység-azonositót viszont nem kötelező.

[p] Opcionális PIP paraméter, amely mind a négy formátumban alkalmazható.

A PIP paraméterek összefoglalása:

[B] Blokkos átvitel előirása

[Dn] Az n-edik pozició utáni összes karakter törlése

[E] A másolás echózása a képernyőn

[F] Lapdobási karakterek szürése

[Gn] File másolása más felhasználói területről

[H] Intel hexa formátum ellenőrzése

[I] A null rekordok ignorálása Intel hexa formátumu átvitelkor

[L] A nagybetük konvertálása kisbetükké

[N] Sorszámozás a listában

[O] Nem ASCII-kódolásu file-ok átvitele

[Pn] Lapdobási karakterek kibocsátása minden n-edik sor után

[Q string CTRL-Z] File egy részének másolása a file elejétől a string első előfordulásáig

[R] Rendszer file másolása

[S string CTRL-Z] File egy részének másolása a string első előfordulásától a file végéig

[Tn] TAB karakterek helyettesitése szóközökkel

[U] Kisbetük konvertálása nagybetükké

[V] A másolati file ellenőrzése /verifikálása/

[W] Csak olvasható /R/O/ file-ok irásának engedélyezése

[Z] A paritásbit törlése az átvitel során

dev:
d:file-név A 3. formátum használata esetén a felhasználónak egy eszköznevet /pl. CON:/ vagy egy file-nevet kell megadnia, ez utóbbi tartalmazhat lemezegység-azonositót is.

Leirás:

1. formátum: Ha a bal oldalon csak d: szerepel, a másolat neve meg fog egyezni az eredeti file-névvel. A bal oldalon levő d:-nek különböznie kell a jobb oldalon megadott d:től. Ha jobb oldalon nem szerepel d:, a másolandó file-nak a kurrens lemezen kell lennie. Ha a "newcopy" argumentum szerepel, akkor a másolat neve más lesz, és ekkor nem kötelező a lemezegység azonositó megadása.

2. formátum: Több egymás utáni PIP kiadása esetén célszerü alkalmazni, mivel ilyenkor a végrehajtás után a PIP a memóriában marad, és a * prompttal ismét bejelentkezik. Ha * után CR-t ütünk, a PIP kilép. Néhány paraméterre nézve a PIP másként müködik, ha parancsként /1.formátum/ vagy ha programként /2. formátum/ hivjuk.

3.formátum: A PIP parancsban eszközneveket /dev:/ és file-neveket /d:file-név/ egyaránt alkalmazhatunk. A csak outputot megvalósitani képes eszközt /pl. nyomtató/ nem lehet inputként definiálni és forditva. A kifejezés bal oldala a céleszköz vagy -file, jobb oldala pedig a forráseszköz vagy -file. A felhasználó konkatenálhat /egymáshoz füzhet/ egyetlen output eszközre vagy file-ba irányuló file-okat.

4.formátum: Több file-t is másolhatunk egyetlen lemezre csoportos file-hivatkozás és opcionális lemezegység-azonositó /d:/ megadásával.
A bal oldalon kötelező használni a lemezegység-azonositót. A másolatok neve megegyezik az eredeti file-ok nevével, de másik lemezen vannak. Két azonos nevü file nem lehet ugyanazon a lemezen /ugyanazon a felhasználói területen/.
A PIP-ben használható eszközneveket és a speciális funkciók végrehajtását előiró kulcsszavakat a függelék tartalmazza.

A PIP.COM-ot bármelyik lemezegységről végre lehet hajtani, ha megadjuk a lemezegység azonositóját a PIP előtt /pl. "B:PIP↵ "/. A PIP-et akkor is végre lehet hajtani, ha kivesszük a rendszerlemezt. A rendszerlemez visszatétele után CR-rel kiléphetünk a PIP-ből.

Példák:

A > PIP B:=*.*↵

/A kurrens lemezen levő összes file-t a B lemezre másolja./

A> PIP↵

* FILE2=TEST2↵

/Létrehozza a TEST2 nevü file másolatát FILE2 néven./

* LST:=FILE2↵

/A FILE2 nevü file egy másolatát az LST: eszközre küldi./

* PUN:=NUL:, PROG.ASM, EOF:↵

/A PUN: nevü eszközre /lyukasztó/ 40 db null karaktert, majd a PROG.ASM nevü file-t, végül az end-of-file /file vége/ karaktert küldi ki./

* B:=PROG.ASM↵

/Létrehozza a PROG.ASM egy másolatát a B lemezen, ugyanazzal a névvel./

* ↵

A > PIP FILE2=FILE1 [G2]↵

/Csak 2.2-ben müködik. A 2-es felhasználói

területen levő FILE1 nevü file másolatát hozza létre ugyanazon lemez kurrens felhasználói területén. A [G2] paraméter jelöli, hogy a másolandó file a 2-es felhasználói területen van./

Megjegyzés: További információk a PIP-et részletesen leiró III/1 fejezetben és a függelékben találhatók.

1Ø. REN

CP/M 1.4
CP/M 2.2

Egy file átnevezése
/Rezidens parancs/

Formátum:

REN uj-név=régi-név

Argumentumok:

uj-név	Az uj és a régi file-név, a kiterjesztés-
régi-név	sel együtt kell megadni

Leirás:

A REN parancs átnevezi a "régi-név" nevü file-t "uj-név"-re. A teljes file-nevet meg kell adni, a kiterjesztéssel együtt.

Mivel beépitett parancs, bármikor végrehajtható.

Példák:

A > REN XYZ.COM=XYZ.XXX ↵

/Az XYZ.XXX file neve ezentul XYZ.COM lesz./

11. SAVE

CP/M 1.4
CP/M 2.2

Memóriatartalom mentése lemezes file-ba
/Rezidens parancs/

Formátum:

SAVE p file-név

Argumentum:

p	A mentendő memóriaterület mérete lapokban /1 lap=256 byte/, decimális alakban.
filenév	Az uj lemezes file neve, kiterjesztéssel együtt.

Leirás:

A SAVE parancs a tranziens programterület /TPA/ tartalmát menti ki a 1ØØH cimtől kezdve p lap /p * 256 byte/ hosszuságban. A memóriaterület a megadott file-név alatt lesz a lemezen. Ezt a file-t ezután végre lehet hajtani, vagy müködését nyomon lehet követni /debug/ ha eredetileg végrehajtható program volt a TPA-ban.

A p értékének meghatározásához először az eredeti programot a DDT segitségével be kell tölteni a memóriába. A NEXT cim 1-gyel nagyobb a program végciménél. Ha a NEXT cim ØØ-ra végződik /pl. 2BØØH/, 1-et le kell vonni belőle /2AFFH/, s az igy kapott szám felső byte-ját decimálissá kell konvertálni /esetünkben ez 2AH=42/. Ha nem ØØ a NEXT cim vége, nem kell levonni 1-et, hanem csak konvertálni a felső byte-ot decimálissá.

Mivel a SAVE beépitett parancs, bármikor végrehajtható.

Az 1.4 verzióban nem lehet két egymást követő SAVE-et kiadni ugyanarra a TPA tartalomra, mivel az első SAVE megváltoztatja a TPA tartalmát. A 2.2-ben ez a probléma nem áll fenn.

Példa:

```
A > DDT MINTA.COM↵
    NEXT   PC
    1CØØ   ØØ
    -GØ↵
```

/Mivel a NEXT cim 1CØØ, ebből 1-et le kell vonni, ez 1BFF, a felső byte 1B, ami decimális 27-tel egyenlő./

```
A > SAVE27COPY.COM↵
```

12. STAT

CP/M 1.4
CP/M 2.2

Státusz-információk kiiratása, eszközök hozzárendelése
/STAT.COM/

Formátum:

1. STAT { DEV: / VAL: }

2. STAT gen:=dev:,gen:=dev:,...

3. STAT d:=R/O

4. STAT d: { file-név / csoportos file-hivatkozás }

5. STAT d: { file-név / csoportos file-hivatkozás } { $S / $R/O / $R/W / $SYS / $DIR }

6. STAT d: { DSK: / USR: }

Argumentumok:

DEV: Az 1. formátumban hatására az aktuális eszközhozzárendelések iródnak ki.

VAL: Az 1.4 verzióban megadása esetén kiiródnak a lehetséges eszközhozzárendelések. 2.2-ben emellett még a STAT parancssorok összefoglalása is megjelenik.

gen:=dev: A 2. formátumban a gen: egy logikai eszköznév /CON:, PUN:, RDR: vagy LST:/ a dev: pedig egy fizikai eszköznév /TTY:, CRT:, BAT:, UC1:, PTR:, PTP:, UR1:, UR2:, UP1:, UP2:, LPT: vagy UL1:/.

d:=R/O A 3. formátumban d: egy lemezegység-azonosító, a d:=R/O hatására a d: azonositóju lemez csak olvasható /read only/ lesz. Ha csak d:-t adunk meg, a STAT kiirja az adott lemezen levő szabad byte-ok számát. STAT

hatására kiiródik a kurrens lemez státusza /csak olvasható /R/O/ vagy irható--olvasható /R/W//és a szabad hely mennyisége.

d: {file-név / csoportos filehivatkozás} a 4. formátumban kiiródnak az adott file vagy file-ok jellemzői /méret, hozzáférés stb./.

{$S / $R/O / $R/W / $SYS / $DIR} Az 5. formátumban a $S paraméter hatására /csak a 2.2 verzióban/ több információ iródik ki a file méretéről. A $R/O paraméter csak olvashatóvá teszi a file-t, megakadályozva annak módositását vagy törlését. Ezt érvényteleniti a $R/W argumentum, melynek hatására a file irhatóvá válik. A $SYS "elrejti" a file-t a DIR parancs elől, ezt a $DIR érvényteleniti.

d:{DSK: / USR:} Csak a 2.2 verzióban a DSK: az egész lemez jellemzőit iratja ki /6.formátum/, a kurrens és aktiv felhasználói terület jellemzőit az USR: megadásával kapjuk meg.

Leirás:

A STAT információt nyujt lemezekről és file-okról , valamint általános eszköznevekhez fizikai eszköznéveket rendel. A 2.2 verzióban csak olvashatóvá tehet egy lemezt /3. formátum/ vagy egy file-t /5. formátum/, ezen kivül kiirja a kurrens felhasználói területet és az aktiv felhasználói területeket.

Példák /2.2 verziót feltételezve/:

```
A > STAT↵
A:R/W, SPACE:2K
B:R/O, SPACE:12ØK
A >

A > STAT B:↵
BYTES REMAINING ON B: 17ØK
A

A > STAT PIP.COM$S↵
Size   Recs   Bytes   Ext   Acc   D:Filename.typ.
 55     55     12K     1    R/O   A:PIP.COM
```

13. SUBMIT

CP/M 1.4
CP/M 2.2

Parancs-file végrehajtása
/SUBMIT.COM/

Formátum:

SUBMIT file-név v1 v2 v3...

Argumentum:

file-név — Parancssorokat tartalmazó szövegfile neve, .SUB kiterjesztésünek kell lennie. A .SUB kiterjesztést nem kell megadni.

v1 v2 v3... A :SUB kiterjesztésü file-ban levő formális paramétereket helyettesitő opcionális aktuális paraméterek. A formális paraméterek $1, $2, $3 stb. alakuak, v1 helyettesiti $1-et, v2 $2-et s.i.t.

Leirás:

A SUBMIT program egy $$$.SUB nevü file-t készit, amely végrehajtódik, miután a SUBMIT program lefutott. Ennek során végrehajtódnak a $$$.SUB file parancssorai. A $$$.SUB file felépitésekor a SUBMIT program a $1, $2, $3 stb. formális paramétereket rendre a v1, v2, v3 stb. aktuális paraméterekkel helyettesiti. A parancsokban szereplő file-ok csak akkor hajtódnak végre, ha az A lemezen vannak.

A .SUB kiterjesztésü parancs-file az ED segitségével hozható létre.

Példa:

Legyen a BATCH.SUB nevü file tartalma az alábbi:

```
DIR  $1.*
PIP  $2:=$1.BAK
ERA  $1.BAK
```

A következő parancs segitségével végrehajthatjuk a SUBMIT programot:

A > SUBMIT BATCH PROG B↵

Ekkor a $$$.SUB file a következőket tartalmazza:

```
DIR PROG.*
PIP B:=PROG.BAK
ERA PROG.BAK
```

Amikor a rendszer befejezte a formális paraméterek helyettesitését az aktuális paraméterekkel, végrehajtja az igy kapott parancssorokat.

14. SYSGEN

CP/M 1.4
CP/M 2.2

A CP/M-másolat generálása; a rendszer betöltése a memóriába és/vagy másolat készitése a rendszerlemezről
/SYSGEN.COM/

Formátum:

```
SYSGEN
```

Leirás:

A SYSGEN program inicializál egy rendszerlemezt /föl-irja a rendszert a lemez első két sávjára, és betölti a rendszert a memóriába, majd végrehajtja azt/.

A > SYSGEN↵

begépelésére a rendszer az alábbi szöveget irja ki:

SYSGEN VERSION m.n

SOURCE DRIVE NAME /OR RETURN TO SKIP/

Válaszul meg kell adni a rendszert tartalmazó lemezegység azonositóját, amely általában A. Ha a rendszer már a memóriában van egy MOVCPM eredményeként, CR-t kell ütni.

A rendszer válasza:

SOURCE ON A THEN TYPE RETURN

Ha a rendszert tartalmazó lemez felpörgött, CR-rel kell válaszolni, amire megjelenik:

FUNCTION COMPLETE

jelezve, hogy az olvasási müvelet befejeződött, a rendszer a memóriában van.

DESTINATION DRIVE NAME /OR RETURN TO REBOOT/

üzenet következik, melyre válaszul meg kell adni azt a lemezegységet, amelyre a másolat kerül. Egyetlen CR begépelése esetén a rendszer melegindítást hajt végre.

DESTINATION ON B THEN TYPE RETURN

jelzi, hogy kész a másolat elkészitésére a gép. Ha a másolatot tartalmazó lemez felpörgött, ezt CR-t ütve kell jelezni.

FUNCTION COMPLETE

jelzi, hogy a másolás befejeződött, az uj rendszerlemez használatba vehető.

DESTINATION DRIVE NAME /OR RETURN TO REBOOT/ ↵

hatására a SYSGEN kilép, és visszakapjuk a rendszer promptot:

A>

15. TYPE

CP/M 1.4
CP/M 2.2

Egy file kiiratása a display képernyőjére
/Rezidens parancs/

Formátum:

TYPE { file-név
       csoportos file-hivatkozás }

Argumentum:

file-név	Meg lehet adni egyetlen filenevet /kiterjesztéssel együtt/,
csoportos file-hivatkozás	vagy több file-t definiáló csoportos file-hivatkozást. Lemezegység-azonositó használata megengedett.

Leirás:

A TYPE parancs bármely file tartalmát meg tudja jeleniteni a képernyőn, olvasható formája azonban csak az ASCII szövegfile-oknak /forrás file-ok, .PRN kiterjesztésü file-ok, lista file-ok stb./ lesz. A TYPE beépitett parancs, igy bármikor végrehajtható.

Példák:

A > TYPE *.TXT↵

A > TYPE B:X.PRN↵

16. USER CP/M 2.2

A kurrens felhasználói terület megváltoztatása
/Rezidens parancs/

Formátum:

USER n

Argumentum:

n	A felhasználói terület száma /Ø és 15 között/

Leirás:

Hatására a parancsban megadott sorszámu felhasználói terület lesz a kurrens felhasználói terület. Hideg inditás /cold start/ esetén a Ø-ás lesz a kurrens felhasználói terület /alapértelmezés/.

A felhasználói terület magyarázatát lásd a III/1. fejezet végén.

Példa:

```
A > USER 2↵
A >
```

17. XSUB CP/M 2.2

Parancsfile feldolgozása során végrehajtott program bemeneti paraméterének generálása
/XSUB.COM/

Formátum:

```
XSUB
```

Leirás:

Az XSUB parancs a .SUB kiterjesztésü file-ok belsejében fordulhat elő. Az XSUB végrehajtása után, ha a SUBMIT file további sorai konzol inputot igényelnek, akkor azt nem a felhasználónak kell begépelnie, hanem a SUBMIT file-ból származnak.

Példa:

Legyen a COMM.SUB file a következő felépitésü:

```
RUN PROG.RAM
XSUB
BACKUP
A
B
```

Ha a következő üzenetet gépeljük be:

```
A > SUBMIT COMM↵
```

az alábbi kiirások jelennek meg a display-n:

```
A > RUN PROG.RAM
.
.   A program lefut
.
A > XSUB
/XSUB ACTIVE/
A > BACKUP
DRIVE FOR SOURCE DISKETTE? A
DRIVE FOR DESTINATION DISKETTE? B
PUT BLANK DISKETTE IN B AND PRESS RETURN
WHEN READY ↵
```

A példában a felhasználónak csak a SUBMIT programot kell elinditania, és CR-t ütnie, ha az üres lemez felpörgött a B lemezegységben. A BACKUP program inputjai a COMM.SUB file-ból származnak.

V. R É S Z :

A CP/M OPERÁCIÓS RENDSZER BELÜLRŐL

1. A CP/M RENDSZER FELÉPITÉSE

Ezen fejezet a CP/M operációs rendszer kialakitását irja le. Célja, hogy a felhasználók saját programjaik megirásakor figyelembe tudják venni a rendszer tulajdonságait, a perifériák, a lemezegység irás/olvasás müveleteinek sajátosságait. Megjegyezzük, hogy a fejezetben előforduló abszolut cimek a CP/M 1.4-es, 16K-s verziójára vonatkoznak.

Az operációs rendszer részei:

CCP:	Console Command Processor	- konzolparancs--feldolgozó
BIOS:	Basic Input/Output System	- periféria-kezelő rendszer
BDOS:	Basic Disk Operating System	- lemez-kezelő rendszer
TPA:	Transient Program Area	- tranziens programterület.

CCP: az operációs rendszer, és a felhasználó közötti kommunikációt biztositja. A billentyüzeten beadott parancsokat értelmezi /interpretálja/ és végrehajtja oly módon, hogy a BIOS és a BDOS modulok megfelelő rutinjait hivja meg.

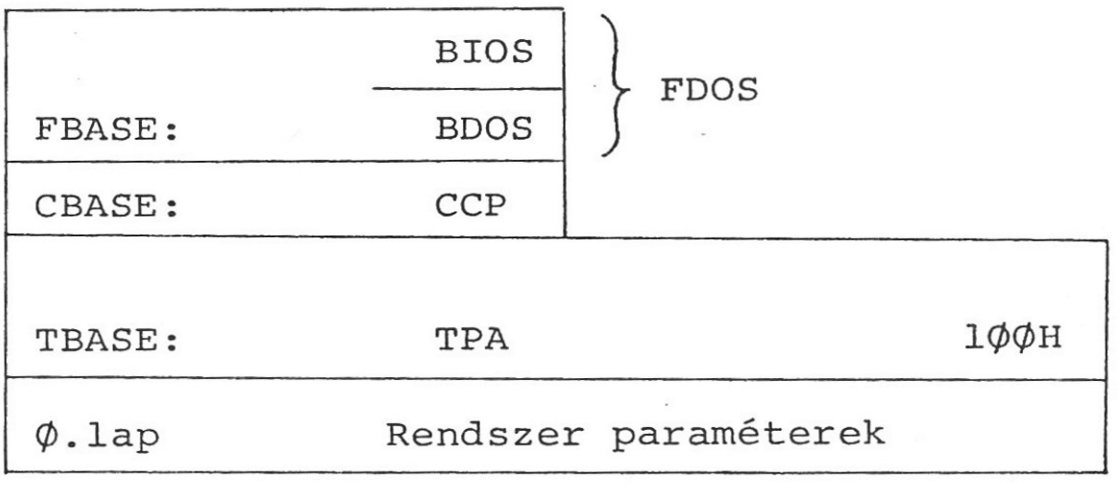
BIOS: az operációs rendszer hardware függő modulja. Ebben a modulban találhatók a perifériák un. driver--rutinjai, amelyek feladata, hogy a perifériák és az operációs rendszer közötti adatforgalmat lebonyolitsák.

BDOS: feladata a lemezes file-ok kezelése. A BIOS és a BDOS közös neve Floppy Disc Operating System /FDOS/.

TPA: A CP/M dokumentációban a programok jellemzésére használják a tranziens /átmeneti/ jelzőt. A TPA /átmeneti programterület/ pedig a memória olyan területe, ahová a különböző, nem állandóan a memóriában tartózkodó felhasználói programok betölthetők és futtathatók. /Minden program a TBASE cimre töltődik be./

A CP/M rendszer 1.4 változatát 16, 32, 48 vagy 64 Kbyte-os méretben lehet generálni. Az ujabb 2.Ø, ill. 2.2 változatban már 1 Kbyte-os lépésekben lehet a rendszert generálni /lásd: MOVCPM/. A legkisebb méretü /16K 1.4-es/ CP/M rendszer esetén 29ØØH /1Ø496D/ cimtartományig helyezkedhet el felhasználói program. A 29ØØH-tól kezdődő terület az operációs rendszeré.

A rendszer elhelyezkedése a memóriában:



TBASE: felhasználói programterület kezdő cime = 1ØØH
CBASE: CCP kezdőcime. 16 K-s rendszer esetén 29ØØH

1.1. A CP/M modulok kapcsolata

A rendszer bekapcsolása és az operációs rendszer betöltése után a CCP veszi át a vezérlést; a display-re kiküldi a "A >" jelzést és várja a kezelői parancsot. Az aktuális parancsbevitelt a BIOS hivásával végzi el, és a parancs sort a CCP pufferje tárolja.

A kezelői utasitás formátuma a következő lehet:

```
UTASITÁS
UTASITÁS FILE-1
UTASITÁS FILE-1 FILE-2
```

Amennyiben a begépelt utasitás un. rezidens parancs, a CCP rögtön végrehajtja azt.

A rezidens parancsok a következők:

```
DIR:  lemez-directory kilistázása
TYPE: ASCII file kilistázása
ERA:  file törlése
REN:  file átnevezése
SAVE: memóriaterület elmentése lemezre
```

Egyes rendszer változatok további rezidens parancsokkal rendelkezhetnek.

Ha a CCP az utasitást nem találja a rezidens parancsok között, akkor azt feltételezi, hogy az utasitás egy tranziens programnak a neve, amely programnévnek

a kiterjesztése ".COM" tipusu /pl. PIP.COM, STAT.COM, ED.COM, stb./.

Ezért utasitja a BDOS modult, hogy keresse meg a file-t, amely a parancsot végrehajtó programot tartalmazza. Ha megtalálta, olvassa be a tranziens programterületre és adja rá vezérlést. /Ezt egy JUMP TBASE utasitás hajtja végre./

Amikor az utasitássorban file-név is található, a CCP egy FILE CONTROL BLOCK-ot /FCB/ képez a file számára. /Az FCB-t részletesen a Rendszerfunkciók c./ alfejezetben ismertetjük./

Ezen FCB alapján betölti a programot és ráadja a vezérlést. Ennek a FCB-nek rögzitett helye van a ϕ. lapon. A cime: 5CH. Ha több nevet adunk meg az utasitás sorban, akkor a második file nevet a 6CH cimtől kezdve teszi el a memóriába mig az összes további nevet a ϕ. lapon lévő pufferban hagyja. /E puffert részletesen lásd a következő fejezetben./ Ilyenkor is csak az első név alapján tölti be a programot. A betöltött programnak kell gondoskodnia arról, hogy a 6CH helyről a file-nevet átirja egy olyan memóriaterületre, amelyet a program használ. A CCP a fenti második és további file nevekhez tartozó file-okhoz nem képez FCB-t.

A FCB fő részei:

- Név terület /11 byte/, ez a file nevet és a kiterjesztést tartalmazza.

- Járulékos információk /8 byte/

- A file-hez tartozó file-területek cime /16 byte/

A CCP az első file-nevet a név területre, a második nevet a blokk foglaltsági területre irja be. A file-kezeléssel kapcsolatos teendőket a BDOS látja el. A BDOS a FCB-ben megadott file hivatkozás alapján azonositja a file-t. A lemezen egy nyilvántartó/directory/ található, amely a lemezen található file-ok nevét, és tartózkodási helyét tárolja. Ezen directory alapján találja meg a BDOS a betöltendő file-okat illetve ebbe a nyilvántartásba tesz bejegyzést, ha ujabb file-t irt fel a lemezre.

A BDOS a FCB-t a directoryban lévő információkkal tölti fel amikor parancsot kap egy file kezelésére.

Amikor a file-t le kell zárni /CLOSE/, a BDOS elvégzi az utolsó szükséges változtatásokat a FCB-ben, és ezt az utolsó állapotot átirja a directory-ba.

Ha a CCP a kezelő legépelt parancsát sem a beépitett parancsok, sem ".COM" file-ként tárolt parancsok között nem találja, akkor a képernyőn megismétli a begépelt parancsot és egy kérdőjelet ir ki utána.

A CCP modulnak nem szükséges állandóan a memóriában tartózkodnia. Egyes programok felhasználhatják ezt a memóriaterületet, akár be is tölthetik magukat a CCP helyére /ezt teszi a DDT is/. Ezeknek a programoknak azonban ugy kell befejezniük müködésüket, hogy az operációs rendszert betöltik. Ez egy JUMP ØØØØH utasitással érhető el.

2. A CP/M BELSŐ VILÁGA

Az eredeti CP/M rendszert Intel MDS-8ØØ mikroszámitógép fejlesztő rendszerre készitették el.

Ugy tervezték meg, hogy a felhasználó a hardware konfiguráció perifériáinak kezelő rutinjait /az un. driver-rutinokat/ megváltoztathassa saját igényének megfelelően. Ilymódon a felhasználó akár olyan rendszert is alkothat, amely IBM-3741 formátumu lemez-vezérlő egységgel egyéb perifériákkal müködik.

Az eszközfüggetlenség elérése érdekében a CP/M három önálló, független modulra van osztva:

BIOS - Basic Input/Output System. A modul tulajdonságai a felhasználó rendszerkörnyezettől függenek. Ahhoz, hogy egy felhasználó a saját mikroszámitógépén futtassa a CP/M operációs rendszert, ezt a modult kell "átszabnia" a saját hardware konfigurációjának megfelelően.

BDOS - Basic Disc Operating System. Ez a modul nem függ a hardware kialakitástól.

CCP - Console Command Processor. Ezen modul kezeli le a felhasználó beadott parancsait, értelmezi, majd a BDOS modul hivásával végrehajtja.

2.1. A CP/M operációs rendszer betöltése

Maga a CP/M operációs rendszer a lemezen található. A számitógép bekapcsolása után ezt be kell tölteni a memória megfelelő területére, és a rendszert inicializálni /alaphelyzetbe állitani/ kell /BOOT/. A rendszer betöltéséhez a CP/M operációs rendszer speciális memóriafelépitést igényel.

A számitógépek bekapcsolása után a programszámláló általában a Ø.cimre mutat, és a processzor attól a cimtől kezdve hajtja végre az utasitásokat. Emiatt a Ø. cimen célszerü állandóan benntartani egy olyan rutint, amely képes az operációs rendszert betölteni. A betöltő program egy részét /kb. 1Ø-2Ø utasitás/ ROM-ba égetve a Ø. cimtől helyezik el. /Megjegyezzük, hogy ettől eltérő megoldások is lehetségesek, pl. a Motorola 68ØØ-as esetében a CP/M FFFFH-ról indirekt ugrással indul./

Ennek a programnak az a feladata, hogy bootstrap módszerrel - többszintü betöltéssel - töltse be az operációs rendszert. E kis program betölti a lemez Ø. track 1. szektoráról a valódi betöltő programot, amely már az operációs rendszert a helyére tölti. Ilymódon a ROM-ban lévő program méretét csökkenteni lehet.

A CP/M operációs rendszer a Ø. cimtől kezdve irható/olvasható /RAM/ memóriát igényel és feltételez. Emiatt a beolvasást követően /amikor a ROM-ban lévő

program lefutott/ a ROM mező lekapcsolódik, és a helyébe Ø. cimtől folytonos RAM terület kerül.

Az operációs rendszer a lemezen az első két sávot foglalja el.

A CP/M a lemezen 128 byte-os szektorokat tételez fel.

2.2. A BIOS ugrótábla

Ahhoz, hogy a korábban emlitett 3 modul /BIOS, BDOS, CCP/ független lehessen egymástól, egy ugrótáblát hoztak létre. Az ugrótábla a driver-rutinok kezdőcimére való feltétlen ugrást biztositja. Minden rutin csak az ugrótáblán keresztül érhető el, a rutinok meghivásához elég ismerni a hivott rutinnak az ugrótáblában elfoglalt belépési pontját. A BIOS-ban találhatók az előre definiált input/output perifériakezelő, valamint a lemezkezelő szubrutinok. Az ugrótáblában az ugró utasitások helye rögzitett, és szükséges, hogy legalább az ugró utasitás megtalálható legyen.

A rutinokat nem kell feltétlenül megirni, elég egy egyszerü "RETURN" utasitás szubrutinként, ha például nem rendelkezünk lyukszalaglyukasztóval.

A BIOS-ról a rendszer többi modulja azt tudja, hol helyezkedik el a WBOOTE cimkére ugró utasitás. Az ugrótábla felépitését lásd a X. függelékben.

Az ugrótábla rutinjait 3 fő tipusra oszthatjuk fel:

- inicializáló /ill. ujrainicializáló/ rutinok:
  BOOT, WBOOT

- karakter input/output rutinok:
  CONST, CONIN, CONOUT, LIST, PUNCH, READER,
  LISTST

- lemez kezelő rutinok:
  HOME, SELDSK, SETTRK, SETSEC, SETDMA, READ
  WRITE, SECTRAN

A felsorolt szubrutinok leirását lásd a fejezetben később.

A CP/M operációs rendszer a perifériákat logikai eszközként kezeli. A BIOS feladata, hogy a logikai eszközre hivatkozó utasitásokat a megfelelő driver-rutin hivására "forditsa le" az ugrótábla segitségével. A karakter I/O müveletek ASCII kódu karakterek kezelését végzik, akár kis-, akár nagy-betüket elfogadva. Egy input eszköz az end-of-file /"EOF"/ jelzést CTRL-Z /1AH/ karakter küldésével jelezheti.

A perifériák fő jellemzői

CON-KONZOL: A konzol logikai eszköznek leggyakrabban CRT /katódsugárcsöves monitor/ vagy teletype felel meg. A konzol müködtetéséhez szükséges handlerek: CONST, CONIN, CONOUT.

LST - LISTÁZÓ eszköz: a fő listázó eszköz egy printer /esetleg teletype/ lehet.

PUN-LYUKASZTÓ: leginkább gyors lyukszalaglyukasztó vagy teletype lehet.

RDR-LYUKSZALAGOLVASÓ

Ezen perifériákat /illetve a perifériához tartozó driver-rutinokat/ a PIP /Peripheral Interchange Program/ használja. Amennyiben a rendszer valamelyik eszközzel nem rendelkezik, a megfelelő driver /LIST, PUNCH, READER/ csak egy "RETURN" utasitásból is állhat, mig a READER rutin az akkumulátor regiszterbe CTRL-Z /1AH/-t téve jelezheti a file végét.

Azt, hogy a fenti logikai perifériáknak mely fizikai perifériák felelhetnek meg, a STAT parancs STAT VAL: változatával rendszerünkön is lekérdezhetjük /ld.STAT leirását/. A fizikai perifériák rövid magyarázata alább következik.

2.3. Az IOBYTE

A CP/M operációs rendszer a felhasználó számára lehetőséget nyujt, hogy logikai és fizikai eszközöket kijelölhessen, szükség esetén megváltoztassa azok korábbi elnevezését. Ehhez a rugalmas változtatáshoz a memória egy rögzitett helyén - a ØØØ3H cimen - egy byte van fenntartva a perifériák számára. Ezt a byte-ot 4 részre osztva 4 fő tipusu perifériát jelöltek ki, mindegyikhez 2 bit-et hozzárendelve.

Ilymódon a byte felosztása a következő:

LIST		PUNCH		READER		KONZOL	
7	6	5	4	3	2	1	ϕ

bitek.

Minden tipusnál a két bit-en 4 különböző eszközt lehet megkülönböztetni. A logikai tipuson belül a fizikai eszközöket jelző kódokhoz a következő perifériák tartoznak:

Konzol:

ϕ - TTY tipusu konzol. Az eszköznév: "TTY:"

1 - CRT tipusu konzol CRT:

2 - BATCH üzemmód: a lyukszalagolvasót konzol input-ként, a listázó eszközt konzol output-ként használja BAT:

3 - A felhasználó által definiált konzol UC1:

Lyukszalagolvasó:

ϕ - A lyukszalagolvasó eszköz egy teletype készülék TTY:

1 - Nagysebességü olvasó az input eszköz RDR:

2 - A felhasználó által definiált olvasó UR1:

3 - A felhasználó által definiált olvasó UR2:

Lyukszalaglyukasztó:

ϕ - Lyukszalaglyukasztóként teletype készülék szolgál TTY:

1 - Nagysebességü lyukasztó PUN:

2 - Felhasználó által definiált lyukasztó UP1:

3 - Felhasználó által definiált lyukasztó UP2:

Listázó eszköz:

Ø - A listázó eszköz egy teletype TTY:

1 - A listázó eszköz egy CRT CRT:

2 - A listázó eszköz egy sornyomtató LPT:

3 - Felhasználó által definiált készülék UL1:

Az IOBYTE megfelelő bitpárjába beirva a megfelelő kódot /Ø-3 között/, kijelölhető a logikai eszköznek megfelelő fizikai eszköz.

A rendszerprogramok közül a PIP.COM és a STAT.COM programok használják az IOBYTE-ot. A PIP adatok és programok mozgatásakor /rendszerbe/-ből/, mig a STAT az IOBYTE által kijelölt logikai-fizikai eszközkijelölés lekérdezését, illetve változtatását teszi lehetővé.

A lemezes irás/olvasási müveletek mindig szubrutin hivási sorozaton keresztül hajtódnak végre. Mielőtt az irási/olvasási parancsot kiadnánk, ki kell választani a megfelelő lemezegységet, be kell állitani a sáv- és szektorszámot, valamint a DMA cimet /annak a memória puffernek a kezdőcime, amelyet a rendszer lemezes I/O-hoz használ/. Mindezen beállitások után hivható meg az irást/olvasást végző rutin. Az iró/olvasó rutinnál bekövetkező hiba esetén 1Ø-szeri rutinhivás

a szokás, mielőtt a rutin hibajelzést adna a hivó programnak.

2.4. A BIOS szubrutinok részletes leirása

BOOT: Ezt a rutint a "hideg start betöltő" /Cold start loader/ hivja meg. A rutin végzi el a rendszer inicializálását /alaphelyzetbe állitását/, szükség esetén az IOBYTE beállitását, valamint üzenetet küld a képernyőre. A rutinnak a WBOOT rutin által beállitott rendszerváltozókat is be kell állitania. A szükséges paraméterbeállitások után a "C" regisztert nullázni kell jelezve, hogy a logikai "A" lemezegységet választja ki, és a vezérlést a CCP-nek kell átadnia→JUMP CCP .

WBOOT: A rutinra melegindítás során kerül a vezérlés. Melegindítás következik be, ha a felhasználói program a ØØØØH cimre ugrik, vagy ha a felhasználó a konzolról CTRL-C paranccsal inicializálja a rendszert. Ezen program feladata az "A" lemezegységben lévő lemez első két sávjáról /track/ beolvasni az operációs rendszert a BIOS kivételével. Ezen felül a Ø. lapon /a memória első 256 byte-ja/ a következő rendszerváltozókat is be kell állitania:

Cim	Tartalom	Megjegyzés
ØØØØH	JUMP WBOOTE	Ugrás a WBOOT belépési pontjára /lásd X. függelék /. Ezen cim alapján kiszámitható az összes BIOS rutin belépési pontja.
ØØØ3H		IOBYTE. Logikai perifériák szükség szerinti beállitása.
ØØØ5H	JUMP BDOS	A tranziens /átmeneti/ programok számára az operációs rendszer elsődleges belépési pontja. Ezen keresztül használhatók a BDOS rutinok.

Az inicializálás befejeztével a rutin a BOOT-hoz hasonlóan a CCP modul kezdőcimére ugrik. A "C" regisztert itt is nullázni kell.

CONST: /console status/, konzol-státusz vizsgálat. Az aktuális konzolt vizsgálja meg, vajon leütöttek-e valamilyen billentyüt. Amennyiben nem, ugy nullát, ellenkező esetben ØFFH-t tölt az "A" /akkumulátor/ regiszterbe és visszatér a hivó rutinba.

CONIN: /console input/; karakter beolvasása a konzolról. A beolvasott karaktert az "A" regiszterben

kell elhelyezni, és a legmagasabb helyiértékü bitet nullázni kell. Amennyiben nincs beolvasható karakter, a hivó rutin a CONIN-ben várakozik.

CONOUT: /console output/; karakter küldése a konzolra. A rutin a "C" regiszterben kapja meg a kiküldendő karaktert. A karakterek ASCII kódban érkeznek. /A legmagasabb helyiértékü bit nullázva van/.

Ebben a rutinban kell illeszteni az operációs rendszert a konzol-terminál speciális tulajdonságaihoz. Ilyen lehet a kontrol-karakterek különleges kezelése, esetleges kiszürése.

LIST: A "C" regiszterből egy ASCII kódu karaktert kiir a listázó eszközre.

PUNCH: A "C" regiszterből egy ASCII kódu karaktert kivisz a lyukasztó eszközre.

READER: Az "A" regiszterbe beolvas egy karaktert az olvasó egységről. A legmagasabb helyiértékü bitnek zérusnak kell lennie. File vége jelzésként a szokásos CTRL-Z jelzést használja a rutin.

HOME: Az aktuálisan kiválasztott lemezegység alaphelyzetbe állitását végzi, az olvasó fejet a ϕ. sávra /track/ állitja. Legegyszerübben egy ϕ. paraméterü SETTRK hivással hajtható végre.

SELDSK: /select disc/; lemezegység kiválasztás.
A "C" regiszterben található a kiválasztandó lemezegység azonositója. Az A logikai egységhez Ø, a B egységhez 1 /és igy tovább/ tartozik, egészen 15-ig /ez a P logikai egységet jelenti/. A standard CP/M rendszer 4 lemezegységes konfigurációval készült el. A lemezegységek számára létrehoztak egy paraméter táblát /disc paraméter table - DPT, részletesen ld. késõbb/. Ezen a 16 byte-os területen az egység jellemzõ paraméterei találhatók. A rutinból való visszatéréskor a HL regiszterpárban a kiválasztott lemezegység paraméter táblájának cimét kell elhelyezni. Amennyiben a felhasználó a rendszerben nem található lemezegységet jelöl ki, ugy a SELDSK a HL regiszterpárba Ø-t irva jelez hibát. Ezt az üzenetet a BDOS kapja meg, és közli a kijelölt output egységen.

A SELDSK rutint célszerü ugy szervezni, hogy a rutin ne forduljon közvetlenül a lemezegységhez amikor meghivják, hanem csak az iró/olvasó rutin hivásakor, hiszen csak akkor szükséges ténylegesen a lemezegységhez fordulni. Elegendõ eltárolni a lemezegység fizikai számát, és majd ezt az iró/olvasó rutin olvassa be, küldi el a lemezegységhez.

SETTRK: /set track/ sávcim /track/ beállitása.
A "BC" regiszterpárban található a standard lemezegység tipusoknál alkalmazott Ø-76 közötti sávcim. A nem szabványos drive-hoz a sávszám Ø-65535 közötti értékeket veheti fel. A

felhasználó választhat, hogy akár ebben a rutinban, akár az aktuális irás/olvasás parancsban küldi ki a lemezegységhez a sávszámot.

SETSEC: /set sector/, szektorszám beállitása.
A "BC" regiszter tartalmazza a megfelelő lemezegységen korábban /SETTRK/ kiválasztott sávon belüli kiválasztandó szektor számot. Hasonlóan az előző rutinhoz a felhasználó választhatja ki, mikor küldi ezt az információt a lemezegységhez.

SETDMA: /set direct memory access address/ a lemezről beolvasott információt tároló puffer kezdőcimének beállitása. Az irás/olvasás müveletek számára a memória puffer /DMA/ cimének beállitását végzi. A "BC" regiszterpár tartalmazza a 8ØH /128D/ byte-os memóriaterület kezdőcimét. Felülirásig minden irás/olvasás művelet a beállitott értéket tekinti puffercimnek.

READ: Olvasási parancs.
Az előzőleg kiválasztott lemezegységen a beállitott sáv és szektorszám és a beállitott DMA-cim szerint beolvas egy szektort. A rutin az "A" regiszterben jelzi vissza a művelet eredményét:

Ø - nem történt hiba, a művelet rendben végrehajtódott

1 - javithatatlan hiba következett be.

Hiba esetén a rutinnak 1Ø-szer meg kell próbálnia

az olvasást. Ha A = 1-gyel tér vissza a rutin az őt meghivó BDOS-hoz, akkor a BDOS hibaüzenetet generál a terminálon:

"BDOS ERR ON X : BAD SECTOR"

X: a lemezegység azonositója.

A hibaüzenet kiirása után a rendszer kezelői beavatkozásra vár. A felhasználó választhat:

- elháritja a hibát /pl. ha nem volt a lemezegységben lemez, betesz egyet/, és a "CR" billentyü leütésével folytatja a müködést,
- CTRL-C billentyü leütésével meghivja a WBOOT rutint, és ujra betölti az operációs rendszert.

WRITE: Egy szektor kiirása a lemezre.
A rutin müködése az adatátvitel irányától eltekintve egyezik a READ rutinéval. /Ezen parancsnál hibaüzenet keletkezhet abban az esetben, ha pl. a lemezegység irásvédelem kapcsoló aktivizálva van, /ha van rajta ilyen/, vagy ha a lemez fizikailag irásvédett, vagy ha a file software uton irásvédett.

LISTST: /status of list device/. A listázó eszköz státuszával tér vissza a rutin az őt hivó programhoz. Az "A" regiszter tartalma Ø, ha a listázó eszköz nem képes karakter fogadására, és ØFFH, amennyiben igen. Ezt a rutint a DESPOOL program használja.

SECTRAN: /sector translation/; logikai szektorszám konvertálása fizikai szektorszámra. Amikor programot, adatot töltünk/ irunk a lemezről / lemezre azon idő alatt a lemez elfordul. A következő szektorra hivatkozó iró/olvasó utasitás már nem a rákövetkező fizikai sorszámu szektort találja. Ha egymás utáni fizikai szektorokba irjuk az adatokat, várni kell tehát, amig a fej alá ujra oda nem ér a megcimzett szektor. Ez nagyon lassitja az adatmozgatást a lemez és a memória között. Ezt a jelenséget ugy küszöbölik ki, hogy a logikai szektorszámlálás és fizikai szektorszámlálás között egy konverziót hajtanak végre. Egy léptető faktorral számitják ki a logikai szektorszámnak megfelelő fizikai szektorszámot. Nézzük meg egy rövid példán:

logikai szektorszám	fizikai szektorszám
1	1
2	7
3	13
4	19
.	.
.	.
.	.

A léptető faktor 6; amig a lemez 6 fizikai szektorral továbbfordul, van idő a memóriába letenni, illetve onnan kiolvasni az adatokat és feltölteni a DMA puffert. Az IBM-kompatibilis, egyszeres sürüségü lemeznél

6-os léptető faktor a megfelelő. A rutin a logikai szektorszámot a "BC" regiszterpárban kapja meg, a transzformáló tábla cimét pedig a "DE" ragiszterpárban. Ez a tábla megegyezik az előbbi példában fizikai szektorszámnak nevezett táblázatrésszel. A rutin feladata, hogy a logikai szektorszámból fizikai szektorszámot számoljon ki, és ezt a "HL" regiszterpárban adja vissza a hivó rutinnak.

2.5. A Ø. lap memória felosztása

Az operativ memóriáról a CP/M leirások lap-nyi mértékegységekben beszélnek /nincs lapozásos memória elérés!/. Egy lap 256 byte /1ØØH/ nagyságu. A lapokat sorszámmal is ellátják. Az operativ memória első lapja a Ø. lap Ø-1ØØH közötti cimek . Ezt a Ø. lapot a rendszer saját paraméterei számára tartja fenn, és csak az 1. laptól felfelé enged tranziens programokat betölteni. Ennek megfelelően a SAVE beépitett parancs az 1-es laptól kezdve 256 byte egészszámu többszörösének megfelelő mértékü memóriát képes a háttértárolóra /lemezre/ elmenteni.

A Ø. lapon található rendszerparaméterek a következők:

Cim -tól	-ig	Memóriatartalom
ØØØØH	ØØØ2H	JUMP WBOOTE utasitás /lásd BIOS ugrótábla/ rögzitett helye. Ezen keresztül

a restart rutin, illetve a BIOS rutinok kezdőcimei kiszámithatóak, elérhetők.

ΦΦΦ3H-ΦΦΦ3H IOBYTE /szerepét lásd korábban/

ΦΦΦ4H-ΦΦΦ4H Kurrens lemezegység. Legutoljára használt lemezegység azonositója található itt.

ΦΦΦ5H-ΦΦΦ7H JUMP BDOS utasitás. Ugrás a BDOS modul belépési pontjára. Utóbbi az a legnagyobb cim, ameddig a tranziens programok használhatják a memóriát. A CCP modul a BDOS modul alatt /alacsonyabb cimeken/ helyezkedik el, azonban egyes programok a CCP-t felülirhatják. Ezt teszi például a rendszerprogramok közül a DDT is.

ΦΦ38H-ΦΦ3AH RESTART 7 utasitás jelentése: ugrás erre a memóriahely cimre. A DDT vagy a SID /szimbolikus debugger/ program használja csak /az operációs rendszer nem/.

ΦΦ4ΦH-ΦΦ4FH A felhasználó által megirt BIOS rutinok számára fenntartott hely.

ΦΦ5CH-ΦΦ7CH A CCP modul ide helyezi a File Control Block-ot /FCB/ .

ØØ7DH-ØØ7FH Random file iras/olvasás müveletek esetén a FCB bővitésére szolgál.

ØØ8ØH-ØØFFH 128 byte-os puffer terület a lemezmüveletekhez. Az inicializálási rutin ezt a cimet állitja be DMA cimként.

Ø1ØØH- Tranziens programok számára fenntartott hely /TPA/ .

2.6. A lemezegységhez tartozó paraméter tábla /Disc parameter táble/ DPT

Ez a tábla, amely szerves része a BIOS-nak, a CP/M operációs rendszer által használt lemezkezelő alaprendszer részletes specifikációját adja. Ezt a specifikációt akár az operációs rendszer alatt futtatható assembler alatt, akár a rendszer alatt futtatható makroassemblerben a DISKDEF makroassembler könyvtár segitségével meg lehet irni.

2.6.1. A DPT fejléc

Minden lemezegységhez egy 16 byte-os un. DPT fejléc tartozik, amelyben a lemezegység kezeléséhez, müködéséhez szükséges paraméterek és pufferterületek cime található. Ezenkivül a BDOS részére átmeneti adatok tárolására van itt hely. A fejléc minden eleme 16 bites szó /cim/.

XLT A logikai szektorszámot fizikaivá transzformáló vektor /tábla/ cime. Ha a fizikai és logikai szektorszám egyezik, a szó értéke Ø. Azok a lemezegységek, amelyek azonos léptető faktorral dolgoznak, ugyanazon cimre hivatkoznak a DPT-ben.

ØØØØ 3 db 16 bites szó. Átmeneti tároló a BDOS számára.

DIRBUF Directory /a lemeznyilvántartó/ pufferének cime. Ez a cim egy 128 byte-os puffert jelöl ki a BDOS számára. Minden lemezegység használhatja ezt a területet. A file kezelési müveletek során használatos ez a puffer.

DPB A lemezegységhez tartozó paraméter blokk cime. Azonos tulajdonságu lemezegységekhez ugyanaz a blokk tartozhat.

CSV Software ellenőrzési terület cime. Minden lemezegységhez külön terület tartozik.

ALV A programok lemezen való elhelyezkedésének ellenőrzési célokra kialakitott térképének cime. Minden lemezegységhez külön térkép tartozik, ezeket a térképeket a BDOS használja.

Minden egyes lemezegységhez tartozik egy fejléc, és e fejlécek összessége a DPT. N db lemezegység esetén a paraméter tábla a következő kialakitásu lesz:

DPBASE:

XLTØØ	ØØØØ	ØØØØ	ØØØØ	DIRBUF	DPBØØ	CSVØØ	ALVØØ
XLTØ1	ØØØØ	ØØØØ	ØØØØ	DIRBUF	DPBØ1	CSVØ1	ALVØ1
XLTn-1	ØØØØ	ØØØØ	ØØØØ	DIRBUF	DPBn-1	CSVn-1	ALVn-1

A DPBASE cimen keresztül érhetőek el a DPT-t alkotó egyes fejlécek, a fejléc minden eleme és egyes fejlécekben található cimeken keresztül a paraméterek.

A BIOS rutinok közül a SELDSK rutin feladata, hogy a hivó rutinhoz való visszatéréskor a "HL" regiszterpárba helyezze a kiválasztott lemezegységhez tartozó lemez paraméter fejléc sorának cimét. A rendszerben nem létező lemezegység kiválasztása esetén "HL" tartalma Ø lesz.

2.6.2. A DPT fejlécelemek alkotórészei

Vegyük sorra a lemez paraméter fejlécben található elemek értékeit.

XLT: Az XLT vektor bárhol elhelyezkedhet a BIOS-on belül.

DPB: A DPB disc paraméter block -cim egy konstansokból álló vektorra mutat, amely a lemez jellemzőit tartalmazza.

A vektor felépitése /az elem alatti szám a méret byte-okban/

SPT	BSH	BLM	EXM	DSM	DRM	ALØ	AL1	CKS	OFF
2	1	1	1	2	2	1	1	2	2

SPT - /sectors per track/ egy sávon belül a szektorok száma

BSH - /block shift factor/ blokk eltolási faktor. Értékét a felhasználó által használni kivánt blokk, másnéven file-terület határozza meg. A file-terület - másnéven blokk az a legkisebb méret, amelyet a rendszer egy program számára lefoglal. Ha egy ilyenbe a program nem fér be, ennek egész számu többszöröse használatos. Az egy file-hoz tartozó file-területek a lemezen nem feltétlenül folytonosan helyezkednek el. Elhelyezkedésük "térképét" a lemez directory-ja tartalmazza az egyes file-ok leirásánál. A fileterület és blokk szavakat ebben a fejezetben szinonimákként fogjuk használni.

A BSH akár képlet alapján számitható, akár táblázatból kikereshető /táblázatot lásd a következő fejléc-elemnél/.

A képlet: $BLSEC = 2^{BSH}$

ahol BLSEC - egy blokk mérete szektorokban mérve. A blokkméret nem szerepel a DPB-ben /disc parameter block/.

BLM - Értéke szintén képlet, vagy táblázat alapján meghatározható.

Képlet alapján:

$$BLM = BLSEC - 1 = 2^{BSH} - 1$$

A táblázat:

BLS	BSH	BLM
1Ø24	3	7
2Ø48	4	15
4Ø96	5	31
8192	6	63
16384	7	127

BLS a byte-okban számitott blokkméretet jelölő decimális szám.

EXM - /extent mask/. Kiterjesztési maszk, amelyet a lemezen elférő fileterület száma és mérete határoz meg.

EXM értéke táblázatból határozható meg:

BLS	DSM < 256 EXM	DSM ≥ 256 EXM
1Ø24	Ø	N/A nem használt
2Ø48	1	1
4Ø96	3	1
8192	7	3
16384	15	7

DSM - /Disk max./ A felhasználó számára elérhető szabad fileterületek száma -1.

A BLS * DSM+1 szorzat megadja a fizikai lemez teljes adatkapacitását /az operációs rendszer részére fenntartott sávokat nem számitva/.

DRM - /Directory max./ A lemezre felirható maximális "directory entry-k" száma -1, ez is, mint általában minden számlálás, Ø-tól indul, 64 entry esetén DRM értéke 63 lesz.

ALØ, AL1 - Ezen két byte-ot 16 bites szónak értelmezve, a directory részére foglalt területet lehet meghatározni fileterületek számában mérve. ALØ a kisebb helyiértékü byte. A Ø. bittől kezdve minden bit beállitása 1 fileterületnyi helyet foglal le a directory számára.

DRM /az entry-k száma/ alapján meghatározható, hogy mely biteket kell 1-nek választani.

Egy példa:
Ha DRM = 127 /ez 128 nevet jelent/ és BLS 1Ø24, akkor 32 programnevet számolva fileterületenként 32 név * 32 byte = 1Ø24 akkor 4 blokkot /területet/ foglal le a directory. Ezért az ALØ byte 4 magasabb helyiértékü bitjét kell 1-re állitani, tehát ALØ = ØFØH és AL1 = ØØH értékü lesz.

A directory részére foglalt blokkok mennyisége képlettel is számitható /az előbbieket, a

BLSEC-re adott képletet, és azt felhasználva, hogy egy szektor a CP/M rendszerben 128 byte/:

$$\frac{DRM + 1}{2^{BSH + 2}}$$

CKS - /Check vector size/. Directory ellenőrzési vektor mérete. Értéke a következő módon határozható meg: ha a lemez cserélhető lemezes /floppy/ akkor

$$CKS = \frac{DRM + 1}{4}.$$

Ellenkező esetben CKS = $\emptyset$
/a directory rekordokat nem kell ellenőrizni./

OFF - /offset/. Az operációs rendszer, és a lemeznyilvántartó /directory/ számára lefoglalt sávok száma. Másképp fogalmazva a programok és adatok számára fenntartott sávok kezdőcime.

Folytatva a DPT fejléc elemeinek vizsgálatát, szólni kell a CSV és ALV cimekről.

CSV - A software ellenőrzési terület mérete, megegyezik a CKS terület byte-jainak számával.

Ha CKS = $\emptyset$, akkor nem szükséges helyet foglalni CSV számára.

ALV - Ezen terület mérete DSM alapján határozható meg. Mérete: DSM/8 +1.

A XI. sz. függelékben egy példán látható egy 2-lemezes rendszer esetén egy 8 inch-es IBM kompatibilis, egyszeres sürüségü floppy egységhez tartozó DPT, a hozzá tartozó fejlécek, és DPB elemei.

2.7. A DISKDEF /disc definiáló/ markokönyvtár használata

A CP/M 2.Ø változata alatt futtatható MAC makroassembler DISKDEF könyvtári rutinját felhasználva a felhasználó könnyen el tudja végezni a lemez paraméter tábla fejléc, illetve a paraméter blokk generálását.

A lemez definiálása a következő definiciók sorozatát igényli:

```
MACLIB                  DISKDEF

 ...
DISKS                   n
DISKDEF                 Ø, ...
DISKDEF                 1, ...
...
DISKDEF                 n-1, ...
egyéb makrohivások /esetleg/
ENDEF
```

A MACLIB parancs betölti a DISKDEF.LIB nevü file-t a makroassembler belső táblájába. A DISKS utasitásban n a lemezegységek száma, ahol n 1 és 16 közötti szám, amely megfelel az A-tól P-ig terjedő logikai lemezegység

kijelölésnek. A DISKS és DISKDEF makróhivások az előző fejezetben definiált adattáblákat hozzák létre. Ez a BIOS-nak olyan része, amely nem végrehajtódó utasitásokat, hanem adatokat tartalmaz.

Az ENDEF makróhivás a szükséges helyfoglalásokat végzi el.

A DISKDEF makróhivás formátuma a következő:

DISKDEF DN, FSC, LSC, SKF , BLS, DKS, DIR, CKS, OFS,

DN - /Disc number/ logikai egység szám Ø és n-1 között

FSC - /First sector/ az első fizikai szektor sorszáma. A lemezvezérlők vagy Ø-val vagy 1-el kezdik számozni a szektorokat, ezt veszi figyelembe ez a paraméter /értéke Ø vagy 1/.

LSC - /Last sector/ a sávon az utolsó fizikai szektor sorszáma.

SKF - /skip factor/ Léptető faktor a logikai-fizikai szektor transzformáció részére. A makróassembler nem generál transzformációs táblát, ha ez a paraméter hiányzik, vagy egyenlő nullával. Amennyiben a szektorok száma kisebb 256-nál akkor a transzformációs tábla elemeinek mérete 1 byte-os, ellenkező esetben 2 byte-os méretü lesz /lásd a SECTRAN BIOS ugrótábla belépési pontnál mondottakat/.

BLS - /Block size/. Ez a paraméter a lemezegységhez tartozó fileterület méretét adja meg byte-okban. Értéke 1Ø24, 2Ø48, 4Ø96, 8192, vagy 16384 lehet.

DKS - /Disc size/. A lemez méretét adja meg a BLS mértékegységében. Például ha BLS = 1Ø24 és DKS = 25Ø, akkor a lemez mérete 25Ø Kbyte.

DIR - /Directory/. A lemeznyilvántartóba maximálisan felvehető file-nevek számát határozza meg.

CKS - /Checked size/. Az ellenőrzött file-nevek számát mutatja. Amint már szó volt róla, a CP/M operációs rendszer ügyel arra, hogy a felhasználó lemezcsere esetén /floppy-nál/ ne tegye tönkre a lemezt olymódon, hogy felülirja a korábban felvett adatokat, programokat. Amiatt floppy-nál CKS=DIR egyenlőség áll fenn, mig fix lemez esetén CKS=Ø, mivel az ilyen lemezt nem lehet kivenni. Ez biztositja, hogy egy adott egységben lemezt csak meleginditás után lehessen cserélni.

OFS - /Offset/. Sáv eltolási szám. Azt a sávszámot jelöli, ahová programok, adatok felirhatók. Az előző sávokon a rendszer és a filenyilvántartás /directory/ található.

O - Flag, amelynek beállitása azt jelzi a rendszer felé hogy a felhasználó a CP/M 1.4 verziónak megfelelő file felirást kiván. Általában ez a paraméter nem használatos, igy elhagyható a definicióból is.

Ezután nézzük meg egy 4-lemezes rendszer definiálását a makrokönyvtár segitségével:

```
DISKS           4
DISKDEF         Ø, 1, 26, 6, 1Ø24, 243, 64, 64, 2
DISKDEF         1, Ø
DISKDEF         2, Ø
DISKDEF         3, Ø

egyéb makrohivások /esetleg/

ENDEF
```

A DISKDEF egységszám, paraméterblokk formátummal egy már előzőleg definiált paraméterblokkot lehet hozzárendelni az adott lemezegységhez.

Ebben a példában az összes lemezegység ugyanazzal a paraméterblokkal /DPB/ rendelkezik.

A lemezegység tulajdonságai:

- 26 szektor/sáv
- a szektorok számozása 1-től 26-ig tart
- a szektor léptető faktor 6
- a lemezen egy adatblokk mérete 1Ø24 byte
- a lemez mérete 243 blokk
- a lemezen maximálisan 64 fileterületnél többet elfoglaló file-ok 2 vagy több azonos névvel, de különböző directory entry-vel szerepelnek a directoryban/
- az operációs rendszernek és a könyvtárnak az első 3 /Ø-2/ sáv van fenntartva.

A makroassembler n db /lásd DISKS n/ lemez paraméter fejléc-et is generál a DPBASE cimtől kezdve.

```
DPBASE  EQU  $

DPEØ:  DW XLTØ, ØØØØH, ØØØØH, ØØØØH, DIRBUF, DPBØ, CSVØ,
       ALVØ

DPE1:  DW XLTØ, ØØØØH, ØØØØH, ØØØØH, DIRBUF, DPBØ, CSV1,
       ALV1

DPE2:  DW XLTØ, ØØØØH, ØØØØH, ØØØØH, DIRBUF, DPBØ, CSV2,

DPE3:  DW XLTØ, ØØØØH, ØØØØH, ØØØØH, DIRBUF, DPBØ, CSV3,
       ALV3
```

A cimke azonositási célokat szolgál. Az elhelyezési és ellenőrzési vektorok területeit az ENDEF makroutasitás hatására a BIOS kód és ugrótábla utáni cimekre helyezi el a makróforditó.

Amennyiben a szektor léptető faktor nulla /vagy kimarad/, az XLT cim értéke ØØØØH lesz. Ellenkező esetben a makroforditó a megfelelő transzformációs táblát generálja az adott cimen.

Az ENDEF makróhivás hatására a forditó szabad memóriaterületet rendel hozzá a megfelelő cimkékhez. A felhasználónak kell gondoskodnia arról, hogy ez a terület szabad legyen a rendszer betöltése után is.

A lemezegység paramétereket a STAT program segitségével lehet ellenőrizni.

Emlékeztetünk a parancs formájára:

STAT d: DSK :

ahol d = A, ...P a lemezegység azonositója.

A parancs hatására a program következő paramétereket jeleniti meg a képernyőn:

r: rekordkapacitás byte-okban

k: lemezkapacitás Kbyte-ban

e: directory névhez tartozó byte-k száma

c: ellenőrzött directory nevek száma

e: rekordok száma a file egy directory entry-jében

b: rekord/fileterület

S: szektor/sáv

t: foglalt sávok száma /rendszer és directory számára/

Végül nézzük meg mi a STAT program válasza 3 különböző lemezdefinicióra:

DISKDEF Ø,1,58, ,2Ø48, 256, 128, 128, 2

r=4Ø96, k=512, d=128, c=128, e=256, b=16, s=58, t=2

DISKDEF Ø, 1, 58, 2Ø48, 1Ø24, 3ØØ,Ø, 2

r=16384, k=2Ø48, d=3ØØ, c=Ø, e=128, b=16, s=58, t=2

DISKDEF Ø, 1, 58, 16384, 512, 128, 128, 2

r=65536, k=8192, d=128, c=128, e=1Ø24, b=128, s=58, t=2

3. RENDSZERFUNKCIÓK

Ebben a fejezetben azt mutatjuk be, hogy a CP/M rendszer milyen lehetőségeket nyujt az input/output eszközök müködtetéséhez. Az I/O parancsok biztositják, hogy a felhasználók rendszereik sajátosságainak megfelelően alakithassák a CP/M rendszerrel történő kommunikációt.

A CP/M rendszer a perifériákat két nagy csoportra osztja:

- egyszerü I/O berendezésekre;
- lemezegységre.

Az első csoporthoz tartoznak a már konvencionális számitógépi perifériákként számontartott lyukszalag olvasó, lyukasztó, sornyomtató, kezelői klaviatura, illetve display.

A második csoportba tartozó lemezegység a file-kezelés minden lehetőségét biztositja, a megnyitástól az irás-son és olvasáson keresztül a lezárásig, mindezt soros és véletlen hozzáférési lehetőség biztositásával.

A rendszerfunkciókat szubrutinok valósitják meg, ezek I/O adatait a CPU regiszterei közvetitik. A szubrutinok hivásának egyszerüsitése végett indulási cimük egységesen a ØØØ5H cim.
Az egyes funkciókhoz tartozó szubrutinok egy, a funkcióra utaló funkciósorszámot visznek magukkal, ez mindig a "C" regiszterben található. A bemeneti adatot vagy cimet a "DE" regiszterpár tartalmazza.

Ha az eredmény 1 byte-os, akkor az "A" regiszterben tárolódik, mig 2 byte-os eredményekkel a "HL" regiszterpárban tér vissza a rutin. Az eredmények felhasználásának kompatibilitása érdekében minden rendszerfunkciós szubrutinból való visszatérés előtt a rendszer végrehajtja a következő értékadást: A = L és B = H, ahol "A", "B", "H", "L" a CPU regiszterei.

Ezt a regiszter átmentési konvenciót az Intel PL/M rendszerprogramozási nyelve is hasonlóan valósitja meg.

A CP/M rendszerfunkciókat megvalósitó szubrutinok jelentős része lemezkezeléssel kapcsolatos. Igy szükségesnek tartjuk, hogy a CP/M rendszer file kezelését röviden bemutassuk.

A CP/M rendszer által létrehozott file-ok tetszőleges számu rekord-ot tartalmazhatnak.

A file név három részből áll:

- a lemezegység-azonositóból,
- a file-névből, amely 8 karakteres és nem tartalmazhat szóköz karaktert,
- valamint a 3 karakteres kiterjesztésből, amely szintén nem tartalmazhat szóköz karaktert.

A file kiterjesztés meghatározó a file felhasználására és felhasználhatóságára vonatkozóan.

A kiterjesztések közül egyeseket az operátor ad meg a file megkülönböztetése céljából, más esetben a rendszerprogramok szolgáltatnak végeredményként adott kiterjesztésü file-t. Pl. az "ASM" assembler fordító-program eredményként az ".ASM" kiterjesztésü file-okból ".HEX" illetve ".PRN" és ".SYM" file-okat készit.

A file-kiterjesztések rögzitett elnevezései olyan szabályokat adnak meg, amelyek a CP/M rendszer egészének müködése szempontjából fontosak.

A file-ok között a forrásnyelvi file-okban /.BAS, .PLI/ a forrásnyelvi program minden 'sorát' kocsi-vissza-soremelés /carrige return - line feed; CRLF/ karakterek zárják le. Igy a lemez 128 byte-os szektora a forrásnyelvi program több sorát is tartalmazhatja.

A file végét rendszerint a CTRL-Z karakter /1AH/, illetve a "file fizikai vége" jelzi, ahol a fizikai véget a file-hoz tartozó utolsó directory entry-ben levő rekordszám /rc- lásd később/ alapján határozza meg a rendszer.

A ".COM" kiterjesztésü file-oknál csak a "file fizikai vége" jelzi, hogy a file véget ért, mivel a CTRL-Z karakter előfordulhat mint információs byte.

A CP/M operációs rendszerben számmal jelölt 128 byte-os rekordok sorozatából áll egy file. A rekordok számozása Ø-tól 65535-ig történhet, igy maximum 8Mbyte-os file-t lehet megvalósitani.

A logikailag folytonos rekordok a lemezen a helytől függően nem feltétlenül folytonosan helyezkednek el. A CP/M rendszer file-jait 16 Kbyte-os darabokra, ugynevezett fileterületekre /extents/ bontja szét.

A file feldolgozása rendszerint a file megnyitásával kezdődik, ahol a "DE" regiszter az ugynevezett file ellenőrző blokkra /FCB = file control block/ mutat. A tranziens programok rendszerint a CP/M rendszer által biztositott és lefoglalt FCB-területet használják a file-müveletek lebonyolitására. Ez a terület a ØØ5CH cimen kezdődik. A filemüveletek mindegyike egy 128 byte-os pufferrel - un. DMA - dolgozik, melynek a ØØ8ØH cimen biztosit helyet a CP/M rendszer.

A file-ra vonatkozó összes információt az FCB tartalmazza, amely 33 byte hosszuságu soros elérésü file-nál, és a file-lal együtt tárolja a rendszer. Az FCB formátumát, a benne szereplő byte-ok jelentését a következő táblázat adja meg:

byte: ØØ Ø1 Ø2 ... Ø8 Ø9 1Ø 11 12 13 14 15 16 ...
3k 32 33 34 35

tart. dr f1 f2 ... f8 t1 t2 t3 ex s1 s2 rc dØ ...
d15 cr rØ r1 r2

ahol

dr A lemezegység /drive/ kódja /Ø-16/ . A lehetséges értékek jelentése: Ø a legutoljára kijelölt lemezegység kiválasztása, azaz

Ø az a lemezegység, ahonnan a rendszert betöltöttük, és ekkor ez az A egység,

1 az "A" jelü lemezegység kiválasztása

2 a "B" jelü lemezegység kiválasztása

.

.

.

16 a "P" jelü lemezegység kiválasztása

f1...f8 A file nevét tartalmazza ASCII nagybetük formájában, azaz a legnagyobb helyiértékü bit - jelölje ezt MSB - Ø értékü.

t1t2t3 Tartalmazza a file tipusát /kiterjesztését/ ASCII nagybetük/MSB=Ø/. Ha e három byte-ban az MSB bitek mégsem Ø értéküek, akkor jelentésük a következő:

MSB /t1/ = 1 csak olvasható file

MSB /t2/ = 1 SYS file /system file/, ez nem jelenik meg a DIR parancs hatására.

ex A file-t leiró directory entry-k közül az éppen használt sorszámát $Ø < ex < 255$ tartalmazza.

s1,s2 A CP/M lemezes operációs rendszer használja.

rc Rekord számláló az "ex"-en belül. Értéke Ø és 128 között változhat.

dØ...d15 A CP/M rendszer ezen byte-okon tárolja a file-hoz tartozó fileterületek /blokkok / cimét. Egy directory entry-hez 16 blokk tartozhat. Amennyiben a file nagyobb, mint 16 blokk, akkor a rendszer a nevet bemásolja egy következő

directory entry-be és "ex" értékét növeli. A file lezárásakor az aktuális FCB paraméterek értéke kimásolódik a directory entry-be. A blokk méretét a BIOS lemez paraméter blokkjában lehet definiálni. Ez rendszerint egy szektor /általában 128 byte/ többszöröse lehet.

cr — Az aktuális rekord olvasása vagy irása esetén soros file-müveletnél rendszerint a felhasználó nullázza.

rØrlr2 — Opcionális véletlen rekordszám Ø és 65535 között. rØ és rl egy olyan 16 bites számot alkotnak, melynél rØ az alacsony, rl pedig a magasabb helyiértékü byte, r2 pedig a tulcsordulást jelzi.

A CP/M operációs rendszerben minden file önálló FCB-vel rendelkezik, amelyről a programozó dönti el, hogy hol helyezkedik el.

File-müvelet esetén a programozó feladata az is, hogy kitöltse az FCB első 16 byte-ját és nullázza a "cr" mezőt is. Rendszerint a file nevét és tipusát tartalmazzák az FCB 1-től 11-ig terjedő byte-jai a többi byte értéke pedig zérus.

Az FCB az operativ memóriában lévő, a file-hoz tartozó információs terület. A lemezen pedig van egy un. directory, amelyben a lemezen lévő file-ok adatai /név, helyfoglalás, stb./ találhatók. A directory-ban minden file-hoz egy vagy több /lásd az előző

fejezetet/ directory entry tartozik. A directory entry az FCB első 32 byte-ját tartalmazza. File irásakor és lezárásakor az FCB másolódik a megfelelő directory entry-be, mig file olvasásakor a directory entry-ben lévő adatok másolódnak át a FCB-ba.

Az operációs rendszer gondoskodik arról is, hogy a file-nevek alapján a következő részben bemutatott utasitások mindegyike megtalálja a számára szükséges paramétereket.

A CP/M operációs rendszer, ahogy azt már a korábbiakban bemutattuk, ugynevezett rendszerfunkcióju szubrutinok segitségével valósitja meg az input/output müveleteket.

A szubrutinok közös vonása, hogy kezdőcimük az operációs rendszerben fix helyen van ØØØ5H . Input/output paramétereiket a továbbiakban mutatjuk be.

3.1. Rendszerfunkcióju szubrutinok

Ø. rutin: Rendszer inicializálás /System Reset/

Input paraméter:

Regiszter C : ØØH

Output paraméter
visszatéréskor:

A "system reset" rutin a vezérlést a CCP-nek adja.

A CCP kiválasztja és aktivizálja az "A" elnevezésü lemezegységet. A rutin lefutásával ugyanazt a hatást érjük el, mintha közvetlenül a rendszer kezdőcimre ugrattunk volna.

1. rutin: Kezelői klaviatura bemenet /Consol Input/

Input paraméter:

Regiszter C: ØlH

Output paraméter
visszatéréskor:

Regiszter A: ASCII karakter

A kezelői klaviatura bemeneti rutin a következő leütött karaktert olvassa be az operátori konzolról. A grafikus karaktereket, mint a kocsi vissza soremelés, vagy a visszalépés /CTRL-H/ a rendszer visszaküldi a megjelenitő egységhez /display/.

A tabulátor karakter a "CTRL-I" 8 karakteres üres helyet hagy. A "CTRL-S" elinditja /megállitja a képernyő információk nyomtatását.

A "CTRL-P" be/kikapcsolást jelent a printerre történő adatkivitelnél.

Ha meghivjuk a kezelői klaviatura bemeneti rutint, az FDOS nem tud visszatérni a rutinból, ameddig egy karaktert le nem ütöttünk a klaviaturán. Igy a további feladatok végrehajtását felfüggesztjük, ameddig a karakter beolvasás nincs készen.

2. rutin: Kezelői display kimenet /Consol-Output/

Input paraméterek:

Regiszter C: Ø2H
E: ASCII karakter

Output paraméter
visszatéréskor:

Az "E" regiszterben elhelyezett ASCII kódu karaktert küldi rutinunk a kezelői kimeneti egységhez /display/. A rutin teljesen hasonló felépitésü az 1. rutinhoz, és az ott leirt tabulátor éskontrol karakterek itt ebben a rutinban is érvényesek.

3. rutin: Olvasói bemenet /Reader Input/

Input paraméter:

Regiszter C: Ø3H

Output paraméter
visszatéréskor:

Regiszter A: ASCII karakter

Az olvasói bemenet rutin a következő karaktert olvassa be a logikai olvasó bemenetről az "A" regiszterbe. A logikai olvasó bemenet definicióját lásd az IOBYTE leirásnál az előző fejezetben.

Ha meghivjuk a rutint, nem adja vissza a vezérlést addig, ameddig a karakter beolvasással nem készült el.

4. rutin: Lyukszalag-lyukasztói kimenet /Punch Output/

Input paraméterek:

Regiszter C: Ø4H
E: ASCII karakter

Output paraméter
visszatéréskor: -

A lyukszalaglyukasztói kimenet egy karaktert küld az "E" regiszterből a logikai lyukasztó egységnek.
A logikai lyukasztó egység definicióját lásd az IOBYTE leirásánál.

5. rutin: Listázási kimenet /List Output/

Input paraméterek:

Regiszter C: Ø5H
E: ASCII karakter

Output paraméter
visszatéréskor: -

A "listázási kimenet" rutin egy ASCII karaktert küld az "E" regiszterből a logikai listázó egységhez.
A logikai listázó egység definicióját lásd az IOBYTE leirásánál.

6. rutin: Közvetlen kezelői bemenet/kimenet /Direct Consol I/O/

Input paraméterek:

Regiszter C: Ø6H
E: ØFFH /inputnál/
vagy ASCII karakter /outputnál/

Output paraméter
visszatéréskor

Regiszter A: karakter vagy státusz

A közvetlen kezelői input/output/ rutint olyankor alkalmazzuk, amikor a kezelői információk minden további jelentéstől független áramlását kell megvalósitanunk. Például a "CTRL-S" vagy "CTRL-P" karaktereknek egyáltalán nem akarunk semmiféle megállitást, illetve listázási állapot jelentést adni.
Mint az input paraméterekből látható, a rutin kétirányu adatforgalmat képes lebonyolitani. Az adatforgalom irányát az "E" regiszter tartalma szabja meg.

Ha az "E" regiszter ØFFH értéket tartalmaz, akkor a rutin adatbemenet irányu és "A=ØØH" értékkel tér vissza, ha nincs input karakter, ellenkező esetben viszont, ha van "A", tartalmazza a klaviaturáról beadott karakter értékét.

Ha az "E" regiszter értéke nem ØFFH a rutin meghivásakor, akkor az azt jelenti, hogy a rutint adatkivitelre használjuk. Ekkor az "E" regiszter egy ASCII karakter kódját tartalmazza.

7. rutin: Add meg az IOBYTE-ot /Get I/O byte/

Input paraméter:

Regiszter C: Ø7H

Output paraméter
visszatéréskor:

Regiszter A: I/O byte értéke /IOBYTE/

A rutin az "IOBYTE" aktuális értékével tér vissza.

8. rutin: Állitsd be az IOBYTE-ot /Set I/O byte/

Input paraméterek:

Regiszter C: Ø8H
E: I/O byte értéke /IOBYTE/

Output paraméter
visszatéréskor -

A "set I/O byte" rutin megváltoztatja a rendszer "IOBYTE" értékét az "E" regiszterben megadott értékre.

9. rutin: Irj ki egy szöveget /Print String/

Input paraméterek:

Regiszter C: Ø9H
DE: A szöveg kezdőcime

Output paraméter
visszatéréskor: -

A rutin egy karakter sorozatot küld az operátori display-hez, melynek memóriabeli kezdőcimét a "DE" regiszterpár tartalmazza. A karaktersorozat végét a $-jel jelöli,igy ez a karakter nem fordulhat elő a kiirandó karaktersorozatban. A 2. rutinban leirt tabulátor és kontrol karakterek hatása itt is érvényes.

1Ø. rutin: Olvasd az operátori input-puffert /Read Console Puffer/

Input paraméterek:

Regiszter C: ØAH
DE: puffer kezdőcim

Output paraméter visszatéréskor:

A pufferbeli karakterek a DE-vel megadott cimtől kezdve

A rutin a kezelői klaviaturáról érkező utasitás sort helyezi el egy memóriaterületre - pufferbe -, melynek kezdőcimét a "DE" regiszterpárban kell megadnunk. A rutin a sorvégjelre, illetve a tárolóterület tulcsordulása esetén fejezi be müködését.
A tárolóterület a memóriában a következő formáju:

DE:	+Ø	+1	+2	+3	.....+n
	mx	nc	C1	C2	??

ahol "mx" a tárolóban elhelyezhető maximális karakter

mennyisége /1-255/, "nc" a megelőző karakter beolvasások száma /pointer/, amelyet az FDOS állit be.

"??" azt a következő szabad memóriahelyet jelöli, amelyet még nem irt felül a rutin saját karakterével.

Az V. függelékben szereplő kontrol karakterek itt is érvényesek, mint adatbeadást vezérlő karakterek.

11. rutin: Add meg a kezelői klaviatura állapotát /Get Console Status/

Input paraméter:

Regiszter C: ØBH

Output paraméter visszatéréskor:

Regiszter A: A klaviatura állapot.

A rutin megvizsgálja, hogy a kezelő leütött-e egy karaktert a klaviaturán. Ha korábban történt billentyühasználat és egy karakter várakozik jelenleg, hogy kiolvassák, akkor az "A" regiszterben "ØFFH" értékkel tér vissza a rutin. Minden egyéb esetben az "A" regiszter értéke "ØØH" lesz.

12. rutin: A verziószám megadása /Return Version Number/

Input paraméter:

Regiszter : ØCH

Output paraméter
visszatéréskor:

Regiszter HL: a verziószám.

A rutin az éppen futó CP/M operációs rendszer verzió-számát adja meg az "L" regiszterben, mint két decimá-lis számot. Például "L= 2Ø" a 2.Ø verziószámot jelöli.

A "H" regiszter CP/M rendszereknél nulla értékü /H=ØØ/, mig MP/M rendszereknél "H=Ø1".

13. rutin: A lemezegység inicializálása /Reset Disc System/

Input paraméter:

Regiszter C: ØDH

Output paraméter
visszatéréskor: -

A rutin programozottan alapállapotba hozza a rendszer-ben lévő lemezegységeket.

Az összes lemezegység közül csak az "A"nevüt engedi használni és a DMA cimet /lásd 26. rutin/ a 8ØH érték-re állitja be.
Ezt a rutint olyan alkalmazói program használhatja, amely a rendszer ujratöltése nélkül teszi lehetővé a lemez cseréjét.

14. rutin: A lemezegység kiválasztása /Select Disc/

Input paraméterek:

Regiszter C: ØEH
E: a kiválasztott lemezegység száma /Ø ÷ 15/

Output paraméter visszatéréskor: -

A rutin kijelöli az "E" regiszterben megadott számu lemezegységet müködésre. Ha "E=ØØ" ez az A jelü, "E=Ø1" a B jelü, és igy tovább az összes lehetséges lemezegységet kijelölheti rutinunk. A rutin végrehajtása után a keijelölt lemezegység ugynevezett "On-line" állapotba kerül, ami azt jelenti, hogy további I/O müvelet elvégzésére alkalmas. Ha az aktiv állapotban lévő egységben lemezt cserélünk, a rendszer automatikusan "csak olvasható" /read only/ állapotba kerül /lásd a 28. rutint/, hogy megvédje a lemezt az illetéktelen felülirástól.

15. rutin: Nyisd meg a file-t /Open File/

Input paraméterek:

Regiszter C: ØFH
DE: FCB cime

Output paraméter visszatéréskor:

Regiszter A: A direktory kódja

Az "Open File" rutin segitségével nyitjuk meg az aktuális felhasználói kódhoz /lásd 32. rutin/ tartozó, adott nevü file-t. Az FDOS a kijelölt lemezegységen keresi az adott nevü file-t ugy, hogy az "FCB" /File Control Block/ 1 - 11. poziciója közötti karaktereivel hasonlitja össze a file nevét.

Az "FCB" cimét a "DE" regiszterpárban kapja meg a rutin és az "FCB" "s1" paraméterét automatikusan nullázza.
Az általunk parancsban megadott file-név kérdőjel karaktereket is tartalmazhat /3FH/. Ezzel jelezzük, hogy az illető pozición nem vagyunk biztosak a karakter kódjában.
Ha a file-t neve alapján megtalálta a rutin, akkor a lényeges directory információt /lásd az FCB részletes leirását/ átmásolja az FCB dØ-tól dn-ig terjedő byte-jaiba, és lehetővé teszi a file-ba történő irást, illetve a file olvasását.

Egy file-hoz mindig egy "Open file" rutin végrehajtása után férhetünk csak hozzá.

A rutin visszatérési paramétere az ugynevezett "directory kód", melynek értéke Ø és 3 között változik, ha a file megnyitása sikeres volt, illetve ØFFH, ha az adott nevü file nem található.

Az "A" regiszterben található directory kód jelentése a következő. A memóriába beolvasott 128 byte hosszuságu rekordban 4 db 32 byte-os FCB fér el a directoryból. Az "A" regiszterben megadott directory kód azt adja meg, hogy a DMA cimen /DMA cim magyarázatát lásd

26. rutin/ a memóriában lévő rekordban hányadik FCB-vel talált egyezést a rutin, igy elősegiti a további információk kiolvasását.

Ha kérdőjel vagy csillag karakter fordult elő az FCB--ben, akkor a keresést az első, egyéb karaktereiben megegyező file névnél befejezi a rendszer.

Megjegyzendő, hogy a "current record" /"cr"/ paramétert nullázni kell a programban, ha a file-ban lévő rekordokat sorrendben az első rekordtól akarjuk elérni.

16. rutin: Zárd be a file-t /Close File/

Input paraméterek:

Regiszter C: 1ØH
DE: FCB cim

Output paraméter visszatéréskor:

REgiszter A: directory kód

A "Close File" rutin pontosan forditottja az "Open File" rutinnak. Az "FCB" cimét itt is a "DE" regiszterben kell megadni.

A directory kód értéke sikeres file zárás esetén Ø és 3 értékek között lehet, mig sikertelen esetben értéke ØFFH. /Jelentését lásd a 15. rutinnál/.

Egy file-t nem szükséges bezárnunk, ha csak olvastuk a tartalmát.

Az irási parancs után viszont feltétlenül alkalmaznunk kell a "Close File" rutint, hogy a file-ról minden információt beirjunk a lemez directory részébe.

17. rutin: Keresd az elsőt /Search For First/

Input paraméterek:

Regiszter C: 11H
DE: FCB cim

Output paraméter visszatéréskor:

Regiszter A: directory kód

A rutin a "DE" regiszterben megadott cimü FCB-vel rendelkező file-t keres a lemezen.

Ha a keresés sikertelen, nem talált ilyen nevü file-t, ØFFH értékkel tér vissza.

Ellenkező esetben - ha megtalálta a file-t - az "A" regiszterben azt Ø, 1, 2 vagy 3 értékkel jelzi.

Ekkor az előzőleg beállitott DMA cimnél /lásd pl. 26. rutin/ lévő 128 byte-os memória pufferben, ahová ez a rutin a lemezről a directory szektorait olvassa be, a legutoljára vizsgált directory rekordot találjuk. A Ø, 1, 2, 3 értékek az "A" regiszterben azt adják meg, hogy hányadik az a 32 byte-os FCB a pufferben, amelynél a névegyezést találta a rutin. Igy a keresett FCB cime a DMA cim + "A" * 32 cimnél található meg / "A" az A-regiszter tartalma/.

A keresési parancsban megadott file név tartalmazhat kérdőjel, illetve csillag /*/ karaktert /un. joker--karaktereket/ bármely pozicióján, csak a lemez egység nevében nem. Ebben az esetben a rutin az első, a feltételezett kielégitő névtalálat esetén visszatér.

18. rutin: Keresd a következőt /Search For Next/

Input paraméter

Regiszter C: 12H

Output paraméter
visszatéréskor:

Regiszter A: directory kód

A rutin müködése hasonló a 17. rutinéhoz, kivéve, hogy a directory-ban történő keresés a már előzőleg megtalált file névtől kezdődik.
Hasonlóan a 17. rutinhoz a 18. rutin is ØFFH értékkel tér vissza az "A" regiszterben, ha nem talál további directory rekordot a filenév kereséséhez.

19. rutin: Töröld a file-t /Delete File/

Input paraméterek:

Regiszter C: 12H
DE: FCB cim

Output paraméter:

Regiszter A: directory kód

A rutin megszünteti az FCB által jelölt file-t. A file név és a kiterjesztés tartalmazhat speciális karaktereket /kérdőjelet, illetve csillag karaktert/, de a lemezegységet kijelölő karakter nem lehet "?", illetve "*" karakter.

A rutin ØFFH értékkel tér vissza az "A" regiszterben, ha nem talált az FCB-vel jelölt nevü file-t a lemezen, ellenkező esetben az "A" regiszter értéke Ø és 3 között van. Az "A" regiszterben található érték jelentése a 17. rutinnál található részletesebben.

2Ø. rutin: Soros olvasás /Read Sequential/

Input paraméterek:

Regiszter C: 14H
DE: FCB cim

Output paraméter
visszatéréskor:

Regiszter A: directory kód

Az FCB-je cimével megadott, előzőleg az "open" vagy "make" rutinokkal aktivizált file-ból olvas a rutin egy 128 byte-os rekordot és elhelyezi a memóriában a kijelölt DMA cimtől kezdően. A rekordot az "FCB" "cr" paraméterében megadott poziciótól kezdi el olvasni, ezután a "cr" paramétert automatikusan növeli a következő rekord pozició értékére.

Ha a "cr" paraméter "tulcsordul", azaz az adott

16 Kbyte-os file-blokkot a végéig elolvastuk, a következő logikai 16 Kbyte-os blokkot automatikusan megnyitja, a "cr" paraméter értékét zérus értéküre állitja be a következő olvasási parancshoz, és visszatér a hivó programba.
Az "A" regiszterben ØØH értékkel tér vissza a rutin, ha az olvasás sikeres volt, illetve nem nulla értékkel, ha az olvasás valamilyen hiba miatt nem sikerült /például a file végére értünk, tehát nincs több olvasható információ/.

21. rutin: Soros irás /Write Sequential/

Input paraméterek:

Regiszter C: 15H
DE: FCB cim

Output paraméter
visszatéréskor:

Regiszter A: directory kód

Az "FCB" cimmel rendelkező, előzőleg az "open" vagy "make" rutinokkal aktivizált file-ba ir egy 128 byte-os rekordot a memóriából, a kijelölt DMA cimtól a rutin.

A rekordot az "FCB" "cr" paraméterében meghatározott helyre irja a rutin, és gondoskodik a "cr" paraméter növeléséről is. Ha a "cr" paraméter "tulcsordul" a következő 16 kbyte-os blokkot /file-területet/ automatikusan megnyitja és a következő irási parancs

előkészitése érdekében a "cr"-t is nullázza, majd visszatér a hivó rutinba. Az "A" regiszter visszatéréskor ØØH értékü, ha az irási müvelet sikeres volt; ellenkező esetben, ha az irási müvelet sikertelen volt /például megtelt a lemez/, értéke nullától különböző.

22. rutin: Készits file-t /Make File/

Input paraméterek:

Regiszter C: 16H
DE: FCB cim

Output paraméter visszatéréskor:

Regiszter A: directory kód

A "make file" rutin müködése hasonlit az "open file" rutinéhoz, a különbség az, hogy az "FCB"-nek olyan filenevet kell tartalmaznia, amely nem szerepel az aktuális lemezegységben lévő lemez directory-jában.

A rutin hatására az FDOS egy file-t nyit meg és a memóriában ugy állitja be az FCB-t, mint egy üres file-ét. A lemezen uj directory entry erről az FCB-ről csak akkor jön létre, ha a file-t lezárjuk, vagy következő extentet kell neki nyitnunk /lásd a fejezet elejét/.

A programozónak figyelnie kell arra, hogy ne fordulhasson elő file-név duplikáció.

A sikeres file létrehozást a rutin visszatéréskor az "A" regiszter Ø és 3 közötti értékkel jelzi.

Ha nincs több hely a lemezen, és a file-t nem lehet létrehozni, az "A" regiszterben ØFFH értékkel tér vissza a rutin. A "make file" rutin alkalmazásával aktivizáltuk az FCB-t, igy nem szükséges a file megnyitásához az "open file" rutin ismételt hivása. /A file a "make file" rutin lefutása után nyitva van!/

23. rutin: A file nevének megváltoztatása /Rename File/

Input paraméterek:

Regiszter C: 17H
DE: FCB cim

Output paraméter visszatéréskor:

Regiszter A: directory kód

Az FCB cimmel megadott, az FCB-ben dr-től t3-ig specifikált file-nak megfelelő directory entry-kben a file-leirást megváltoztatja az ugyanezen FCB-ben d1-től d11-ig megadott információra.

A lemezegység kódja "dr" az FCB Ø pozíciójánál az egység azonositóját adja meg, mig az uj file névnél lévő "dr" kódnak /dØ/ zérusnak kell lennie. A rutin visszatéréskor Ø és 3 közötti értéket helyez az "A" regiszterbe, ha az "átnevezés" sikeres volt, egyébként az "A" regiszter értéke ØFFH, ha nem találta meg az átnevezendő file-nevet.

24. rutin: A lemezegységek müködésének állapota /Return Login Vector/

Input paraméter:

Regiszter C: 18H

Output paraméter visszatéréskor:

Regiszter HL: állapotvektor

A rutin végrehajtásakor a "HL" regiszterpárban egy 16 bites értéket kapunk, melynél a "L" regiszter LSB bitje az "A" nevü lemezegységgel, a "H" regiszter MSB bitje pedig a 16. lemezegységgel /P/ kapcsolatosan tartalmaz információt.

A bitek "Ø" állapota azt jelenti, hogy az egység nincs "bekapcsolva", mig a bitek "1" állapota jelzi, hogy az egység "on-line" állapotban van.

Az "on-line" állapot azt jelenti egy disc egységnél, hogy az egység kiválasztása a korábbiakban megtörtént. A korábbi CP/M verziókkal való kompatibilitás végett az "A" és az "L" regiszter ugyanazon értéket tartalmazza a rutinból történő visszatéréskor.

25. rutin: A kurrens lemezegység neve /Return Current Disc/

Input paraméter:

Regiszter C: 19H

Output paraméter
visszatéréskor:

Regiszter A: a kurrens lemezegység száma.

A rutin futása után megadja az "A" regiszterben az éppen müködő lemezegység számát. A lemezegység száma Ø és 15 között változhat, ezek a számok rendre az A-tól a P-ig terjedő egységeknek felelnek meg.

26. rutin: Állitsd be a DMA cimet /Set DMA Address/

Input paraméterek:

Regiszter C: lAH
DE: DMA cim

Output paraméter
visszatéréskor: -

A "DMA" a "Direct Memory Access" röviditése, amely közvetlen memória hozzáférést jelent. A fogalom, illetve használata a disc egység vezérlő részének illesztésénél játszik szerepet, és az adatforgalom módját adja meg.
Azon tul, hogy a számitógép és a lemezegység között programvezérelt I/O csatornán is meg lehet valósitani az adatforgalmat, a CP/M a lehető leggyorsabb közvetlen memóriahozzáféréses adatforgalom lehetőségét is biztositja. Ez azt jelenti, hogy a rutinok a lemez tartalmához nem byte-onként, hanem egy memóriapuffernyi - ez a CP/M-nél 128 byte - egységenként jutnak hozzá. E puffer cimét, akárcsak a fenti hardware fogalmat, DMA-val jelöljük, azaz a DMA cim azt a helyet jelenti, ahol a lemezre irás előtt, ill. beolvasás

után a 128 byte-os rekord helyet
foglal. Hideg, illetve melegindítás /"cold" vagy "warm start"/ vagy a lemezegység inicializálása /"reset"/ után a DMA cimet a rendszer automatikusan 8ØH értékre állitja. A rutin segitségével lehetőségünk van ezt a kezdőértéket megváltoztatni.

A DMA cim értéke igy az ujonnan beállitott érték lesz egy ujabb cimet tartalmazó "Set DMA" hivásig, hideg vagy melegindításig, illetve rendszer inicializáló rutin hivásáig.

27. rutin: A lemezegység információs vektorának cime /Get Address of Allocation Vector/

Input paraméter:

Regiszter C: 1BH

Output paraméter
visszatéréskor:

Regiszter HL: Az információs vektor cime

A CP/M rendszer egy ugynevezett "információs vektort" készit minden müködésben lévő "on-line" állapotu lemezegységről.

Különböző rendszeprogramok, mint például a STAT program, felhasználják az "információs vektor" tartalmát. A rutin végrehajtása után a "HL" regiszterben a kurrens lemezegység információs vektorának cime található.

Ezt a rutint alkalmazói programok normális körülmények között ritkán alkalmazzák.

28. rutin: Lemezegység irásvédelme /Write Protect Disc/

Input paraméter:

Regiszter C: 1CH

Output paraméter
visszatéréskor: -

A lemezegység irásvédelem rutin az aktuálisan kiválasztott lemezegység átmeneti irásvédelmét valósitja meg. A rutin végrehajtása után irási parancsot nem hajt végre a rendszer az illető lemezegységre.

Irási kisérlet esetén a következő üzenetet küldi az operátori display-re:

Bdos Err on d : R/O

ahol "d" a lemezegység azonositója /A-P/. Az irásvédelmet a következő hideg, illetve melegindítás szünteti meg.

29. rutin: Add meg a csak olvasható lemezegységeket /Get Read Only Vector/

Input paraméter:

Regiszter C: 1DH

Output paraméter
visszatéréskor:

Regiszter HL: A read only tulajdonságot jelző bitek vektora

A rutin végrehajtása után megadja a "HL" regiszterpárban az átmenetileg irásvédett lemezegységek bit vektorát. Az egyes bitek jelentése: "Ø" nincs irásvédettség, "1" az adott bit-hez tartozó lemezegység irásvédett.

A lemez irás elleni védettsége két módon jöhet létre, az egyik a 28. rutin meghivásával, a másik, amikor egy automatikus irásvédelmi mechanizmus lép életbe a CP/M rendszerben, amely érzékelte, hogy lemezt cseréltünk egy egységben, a rendszer inicializálása nélkül.

30. rutin: Állitsd be a file-információkat /Set File Attributes/

Input paraméterek:

Regiszter C: 1EH
DE: FCB cim

Output paraméter
visszatéréskor:

Regiszter A: directory kód

A rutin lehetőséget ad arra, hogy a file-hoz tartozó állandó paramétereket programozottan megváltoztathassuk. Ilyen paraméterek az irásvédettség, illetve, hogy valamely file ugynevezett "SYS" /System/ file. Ezeket az információkat a file-név kiterjesztést adó byte-ok MSB bitje jelzi /lásd a fejezet elejét/. Ha az FCB "t1" karakterénél az MSB bit=1 a file irásvédett /csak olvasható/. Ha az FCB "t2" karakterénél az MSB bit=1, az a file ugynevezett "system file", például a "DIR" parancs esetén nem jelenik meg a listában.

Az FCB "t3" karakterének legnagyobb helyiértékü bitje a rendszer további bővitésére van fenntartva és nincs jelentése.

A "DE" regiszterpár azon "FCB" cimét tartalmazza, amelyben adott jellemzőkkel a file-t el akarjuk látni. A rutin megkeresi a lemezen az ezen FCB-ben adott névhez tartozó directory rekordot és átirja a benne szereplő paramétereket.

31. rutin: A lemezegység paraméterei cimének beolvasása /Get Address of Disc Parameters/

Input paraméterek:

Regiszter C: 1F^{H}

Output paraméter:

Regiszter HL: a lemezegység paraméterblokkjának cime

A rutin végrehajtása után a "HL" regiszterpárban megadja a "BIOS"-beli lemez paraméter tábla /lásd BIOS leirás kezdőcimét - DPBASE/ cimét.

Az alkalmazások többségében ezt a rutint felhasználói programok egyáltalán nem használják.

32. rutin: A felhasználó kód beállitása/lekérdezése /Set/Get User Code/

Input paraméterek:

Regiszter C: 2ØF
E: ØFF - lekérdezés
felhasználói kód-beállitás

Output paraméter:

Regiszter A: Az aktuális felhasználói kód /lekérdezésnél/
nincs értéke /beállitásnál/

A rutin lekérdezi, illetve beállitja az aktuálisan aktiv felhasználói kódot. Ha az "E" regiszter ØFFH, akkor az aktuális felhasználói kóddal tér vissza az "A" regiszterben, amely Ø és 31 közötti értéket vehet fel.

Ha az "E" regiszter értéke nem ØFFH, akkor az aktuális felhasználói kódot megváltoztatja az "E" regiszterben található értékre /modulo 32/.

33. rutin: Közvetlen olvasás /Read Random/

Input paraméterek:

Regiszter C: 21H
DE: FCB cime

Output paraméter
visszatéréskor:

Regiszter A: Hiba kód

A rutin müködése hasonló a "soros olvasás" rutin müködéséhez, azzal a különbséggel, hogy az olvasandó rekord paraméterei egy speciális 24 bites regiszterben vannak /az "FCB" "rØ", "r1" és "r2" byte-jaiban/. Az "rØ", "r1", "r2" tárolási módja: "rØ" a legkevésbé értékes helyiértékü byte, "r1" a középső byte és "r2" a legmagasabb helyiértékü byte.

Az "rØ" és "r1" byte párból képzett 16 bites szó tartalmazza az olvasandó rekord számát /lásd a fejezet elején az FCB kifejtését/. Ez az érték Ø-tól 65535-ig változhat, igy a 8 Mbyte memóriaterület elérésére van lehetőségünk 128 byte-os rekordokkal.

A közvetlen olvasási módnál a CP/M rendszer az adott file-hoz tartozó 16 kbyte-os file-területek közül megkeresi, hogy melyikben van az adott rekord számhoz tartozó rekord és elvégzi a beolvasást. Az olvasás eredményességét a rendszer az A regiszterben jelzi vissza. A hibakódok jelentése:

Az "A" regiszter tartalma	Jelentése
∅	Az olvasási müvelet hibátlanul megtörtént.
1	Olyan helyről olvas, ahová korábban nem irtunk semmit.
2	Ez a kód nem fordulhat elő közvetlen olvasásnál.
3	Nem képes az olvasási parancs bezárni az aktuális file-területet a lemezen.
4	Felüliratlan file-területet keres a rutin.
5	Ez a kód nem fordulhat elő közvetlen olvasásnál.
6	Keresés a lemez fizikai végén tul.

Az 1 és 4-es hibák akkor jelentkeznek, amikor olyan adatokat akarunk olvasni, amelyeket előzőleg nem irtunk fel a lemezre.

34. rutin: Közvetlen irás /Write RAndom/

Input paraméterek:

Regiszter C: 22H
DE: FCB cime

Output paraméter
visszatéréskor:

Regiszter A: Hibakód

A "közvetlen irás" rutint hasonlóan kell előkésziteni, mint a "közvetlen olvasás" rutint, a különbség egyedül az, hogy az adatok kiirását a lemezre az aktuális DMA cimtől kezdi el a rutin. Ha egy file-terület vagy adat blokk, amelybe irni szeretnénk, még nem létezik a rutin hivásakor, a rutin végrehajtja a helyfoglalást.

Ugyanugy, mint a "közvetlen olvasási" rutinnál a rekord szám nem változhat meg az irási parancs végrehajtása közben.
A hibakód visszatéréskor a "közvetlen olvasásnál" leirtakkal megegyezik, hozzátéve azt, hogy az "5" hibaüzenetet a lemez teli állapota is előidézheti.

35. rutin: A file méretének számitása /Compute file size/

Input paraméterek:

Regiszter C: 23 H
DE: FCB cime

Output paraméterek
visszatéréskor:

A közvetlen hozzáféréshez tartozó paraméterek "rØ", "r1", "r2"

A rutin az "FCB"-je cimével megadott file-t megkeresi a lemez directory-ban és visszatéréskor a közvetlen hozzáférés paramétereiként szereplő byte-ok a "virtuális" file-méretet tartalmazzák, amely annak a rekordnak a száma, amely az "end of file" /file vége/ rekordot követné.

Ha a rutin hivása után az "r2" paraméter értéke "1", ez azt jelenti, hogy az "rØ" "r1" paraméter byte-okból képzett 16 bites szám a maximális értéket, 65535-öt tartalmaz. Egyébként az "rØ", "r1" byte-pár egy 16 bites értéket tartalmaz, amely a file un. virtuális mérete.

A file virtuális mérete megegyezik fizikai méretével soros file-kezelés esetén.

Nem ez a helyzet közvetlen irási módnál, ahol "lyukak" maradnak a file-ban, és ehhez kell file-méretet számitanunk. Tény, hogy ebben az esetben a rutin által kiszámitott file-méret és a ténylegesen használt rekordok összmérete között lényeges eltérés van.

36. rutin: Állits be közvetlen rekordelérést /Set Random Record/

Input paraméterek:

Regiszter C: 24H
DE: FCB cime

Output paraméter
visszatéréskor:

A kurrens rekord mező beállitása.

A rutin végrehajtásakor utasitja a "BDOS"-t, hogy automatikusan hozzon létre egy "kurrens rekord poziciót" /rekordszám / ahhoz a file-hoz, amelyet éppen most olvasunk vagy irunk soros módon. A rutin felhasználására a következő példát adhatjuk meg. Egy sorosan olvasott file-ban egy adott ponttól /adattól/ kezdődően közvetlen szeretnénk a file-t olvasni, mert algoritmusunk ezt kivánja meg. A rutin meghivása után az adott file-t közvetlen módon olvashatjuk /irhatjuk/ attól a rekordtól kezdődően, ahol a soros/közvetlen váltás megtörtént. Természetesen a közvetlen olvasás/irás azokra a rekordokra is vonatkozik, amelyek az "átváltás" előtti pozición helyezkednek el.

Rendszerfunkcióju rutinok összefoglaló táblázata

Szám	N é v	Bemeneti paraméter	Kimeneti eredmény
ϕ	Rendszer inicializálás	-	-
1.	Kezelői klaviatura input	-	A=karakter
2.	Kezelői display output	E=karakter	-
3.	Olvasói input	-	A=karakter

4.	Lyukszalag lyukasztó output	E=karakter	-
5.	Lista output	E=karakter	-
6.	Közvetlen kezelői input/output/	E=jelzőszám/ /karakter	karakter v. státusz
7.	Add meg az IOBYTE-ot	-	A = IOBYTE
8.	Állitsd be az IOBYTE-ot	E=IOBYTE	-
9.	Irj ki szöveget	DE=Szöveg cim	-
10.	Olvasd az operátori input puffert	DE=puffer-cim	puffer tartalma
11.	Add meg a kezelői klaviatura állapotát	-	A=ØØH/FFH
12.	Verzió szám megadása	-	HL= verzió
13.	A lemezegység inicializálása	-	-
14.	A lemezegység kiválasztása	E=disc száma	-
15.	Nyiss meg file-t	DE=FCB cim	A=DIR.kód
16.	Zárd be a file-t	DE=FCB cim	A=DIR.kód

17.	Keresd az elsőt	DE=FCB cim	A=DIR.kód
18.	Keresd a következőt	-	A=DIR.kód
19.	Töröld a file-t	DE=FCB cim	A=DIR.kód
20.	Soros olvasás	DE=FCB cim	A=DIR.kód
21.	Soros irás	DE=FCB cim	A=DIR.kód
22.	Készits file-t	DE=FCB cim	A=DIR.kód
23.	A file nevének megváltoztatása	DE=FCB cim	A=DIR.kód
24.	A lemezegységek müködésének állapota	-	HL=Müködési vektor
25.	A kurrens lemezegység száma	-	A=lemezegység száma
26.	Állitsd be a DMA cimet	DE=DMA cim	-
27.	A lemezegység információs vektorának cime	-	HL=Inf.vektor cime
28.	Lemezegység irásvédelme	-	-
29.	Add meg a csak olvasható lemezegységeket.	-	HL=csak olv.lemezegységek vektora
30.	Állitsd be a file információkat	DE=FCB cim	A=DIR.kód

31.	A lemezegység paramétereit megadó blokk	-	HL=Blokk cim
32.	A felhasználói kód beállitása/kérdezése	lásd a rutinnál	
33.	Közvetlen olvasás	DE=FCB cim	A=Hiba kód
34.	Közvetlen irás	DE=FCB cim	A=Hiba kód
35.	A file méretének számitása	DE=FCB cim	rØ rl r2
36.	Állits be közvetlen rekord elérést	DE=FCB cim	rØ rl r2

4. RENDSZERGENERÁLÁS: A CP/M INSTALLÁLÁSA ÉS MÓDOSITÁSA

A CP/M operációs rendszer üzembe helyezésekor a fő problémát az okozza, hogy a felhasználó perifériái általában nem azonosak a standard CP/M által feltételezett hardware környezettel /MDS-8ØØ/. A CP/M három alapvető része /CCP, BDOS, BIOS/ közül csak a BIOS /alapvető input-output rendszer/ függ a hardware környezettől. Első lépésként tehát létre kell hozni a felhasználó saját input-output kezelő programját /CBIOS/. Ennek megirása és tesztelése több lépésben

történik, és függ attól, hogy milyen segédeszközök állnak rendelkezésre. Ha az installálás a CP/M egy korábbi verziója alatt, vagy egy más hardware környezetü, de szintén CP/M-et használó mikroszámitógépen történik, akkor alkalmazhatók a CP/M tranziens parancsai /MOVCPM és SYSGEN/. Ez a fajta installálás második szintü rendszergenerálás néven foglalható össze, ismertetésére a későbbiekben kerül sor. Ha a CP/M installálása nem egy másik CP/M operációs rendszer alatt történik, akkor minimális fejlesztői segédeszközökkel /EDITOR, ASSEMBLER, LOADER/ kell megirni a CP/M-et az eredeti rendszerlemezről beolvasó /GETSYS/ és oda kiiró /PUTSYS/ programot. A visszairás előtt természetesen az eredeti, MDS-8ØØ-at kezelő BIOS-t helyettesiteni kell a saját gép perifériáit kezelő CBIOS programmal. A fentebb vázolt második installálási lehetőséget nevezik elsőszintü rendszergenerálásnak. A továbbiakban ismertetendő információk a CP/M 2.Ø verziójára vonatkoznak. Ez főként a memória-cimekben jelent eltérést az 1.4 verzióhoz képest.

A standard CP/M 2.Ø 2Ø Kbyte RAM memóriát tételez fel. Nagyobb memória esetén egy b cimeltolás adandó a szereplő cimekhez, amely a memóriaméret - 2ØK képlet segitségével számitható ki. b értéke néhány szokásos memórianagyság esetén /a H betü a számok végén a hexadecimális számrendszerre utal/:

24K : b=1ØØØH
32K : b=3ØØØH
48K : b=7ØØØH
64K : b=BØØØH

Először a 2ØK-s rendszert kell létrehozni, majd

azt átkonfigurálni a tényleges memóriaméretre.

4.1. Elsőszintü rendszergenerálás: GETSYS, PUTSYS

Az elsőszintü rendszergenerálás első lépéseként meg kell irni egy GETSYS nevü programot, amely beolvassa az eredeti rendszerlemez első két sávját a memóriába a 338ØH cimtől kezdve. A GETSYS programnak a 1ØØH cimen kell kezdődnie /a TPA eleje/.

A GETSYS program egy lehetséges változatának leirása a függelékben található. A GETSYS tesztelése ugy történhet, hogy egy üres lemez első két sávját olvassuk be a memóriába, majd ellenőrizzük, hogy sikerült-e a beolvasás. Ha a tesztelés sikeres, beolvashatunk egy rendszert tartalmazó lemezt a memóriába a GETSYS segitségével.

Ezután egy PUTSYS nevü program megirása következik, amely a memória tartalmát szintén a 338ØH cimtől kezdődően, felirja a lemez első két sávjára. A PUTSYS program kezdőcime 2ØØH kell, hogy legyen. A PUTSYS-t oly módon tesztelhetjük, hogy segitségével egy üres lemez első két sávjára felirjuk a memória egy részének ismert tartalmát, majd ezt visszaolvassuk a már megirt GETSYS segitségével, és ellenőrizzük az egyezést.

Ha a GETSYS és a PUTSYS elkészült, /1-1 mintaprogramot lásd a függelékben/ meg kell irni a saját perifériakészlet kezelését elvégző BIOS programot

/CBIOS/. Ebben először csak primitiv lemezmüveleteket és egyszerü konzol input-outputot kell megvalósitani. A CBIOS tesztelésekor arról kell meggyőződni leginkább, hogy a lemez olvasása során nincs-e véletlenszerü irás a diskre, és hogy a sáv- és szektor-cimzés megfelelő-e.

A CBIOS elkészültével be kell olvasni a CP/M rendszert a lemezről, a BIOS a 4AØØH-4FFFH cimtartományban lesz. E területre kell átvinni a CBIOS-t, majd a PUTSYS programmal ki kell menteni a módositott rendszert a lemezre.

Ezután a GETSYS segitségével vissza kell olvasni a módositott rendszert a memóriába a 338ØH cimtől /CBASE - a bootstrap loader program kezdőcime, 2.Ø CP/M verzió esetén; ez az 1.4-es verziónál 288ØH/ kezdődően. A betöltés ellenőrzése után a hideg inditás cimére lehet ugrani, ez 4AØØH /lásd a BIOS ugrótábla felépitését megadó függeléket /. A hideg inditás inicializálja a Ø-ás lapot, majd a CCP-be ugrik /34ØØH/, amely meghivja a BDOS-t, ez viszont hivja a CBIOS-t. A CCP kérésére a CBIOS be fog olvasni 16 szektort a 2-es sávról, s ha ez sikerül, a CP/M bejelentkezik az "A>" rendszer--prompt-tal. Az uj rendszer tesztelésére először ki kell adni egy

SAVE 1 X.COM↵

parancsot. Ennek hatására számos lemezes átvitel lezajlása után a CP/M ismét bejelentkezik:

A>

A directory parancs tesztelésére a

DIR↵

parancsot kell kiadni, erre meg kell jelennie az

A: X.COM

directorynak.

A törlés végrehajtásának vizsgálata az

ERA X.COM↵

parancs begépelésével lehetséges, amelyre a rendszernek ki kell törölnie a file-t és vissza kell adnia a promptot.

A következő lépés egy kezdeti betöltő program /bootstrap loader/ megirása, amely a Ø-ás sáv 1-es szektorára kerül. Hideg inditáskor ez a kezdeti betöltő program tölti be a lemez első két sávját /magát a CP/M rendszert/ a memóriába a 34ØØH+b /CBASE+b/ cimtől kezdődően. A betöltés a Ø-ás sáv 2-es szektorával kezdődik, önmagát ugyanis természetesen nem tölti be. A kezdeti betöltő tesztelése után - ha az helyesen müködik - kipróbálhatjuk a meleg indítást a CTRL és a C billentyü egyidejü lenyomásával /CTRL-C/. A rendszernek ekkor ismét be kell jelentkeznie. Az installálásnak ebben a fázisában a generált rendszer valószinüleg hibátlan, igy - a GETSYS és a PUTSYS segitségével - átmásolható a rendszerlemezre. Ettől kezdve, ha lemezcserére kerül

sor, mindig ujra kell tölteni a rendszert CTRL-C leütésével, máskülönben az ujonnan behelyezett lemezre nem lehet irni. Ezután hasznos egy COPY nevü program létrehozása, amely lemezek teljes tartalmának másolására szolgál. Ennek segitségével másolatokat hozhatunk létre a rendszerlemezről arra az esetre, ha az eredeti példány valamilyen okból használhatatlanná válik.

Az első szintü rendszergenerálás utolsó fázisa a CBIOS bővitése a rendelkezésre álló további perifériák /szalaglyukasztó, - olvasó, stb./ kezelésére. Ez elvégezhető a GETSYS és a PUTSYS segitségével, vagy a következőkben ismertetendő második szintü rendszergenerálás előirásait követve.

4.2. Második szintü rendszergenerálás: MOVCPM

A második szintü rendszergenerálás lényege, hogy a CP/M-et alkalmassá tegyük a teljes felhasználói memória nyujtotta lehetőségek kiaknázására. Ez a MOVCPM /rendszer-relokátor/ program futtatásával lehetséges:

A>MOVCPM bb*↵

ahol bb a memória mérete kilobyte-okban, decimálisan, /pl. 32, ha a memória mérete 32 Kbyte/. A rendszer válasza:

CONSTRUCTING bbK CP/M VERS m.n

```
READY FOR "SYSGEN" OR
"SAVE 34 CPMbb.COM"
```

Ekkor a CP/M-nek a megadott memóriaméretre konfigurált változata a TPA-ban van a 9ØØH-227FH területen, a memóriaképben levő BIOS és BOOT /kezdeti betöltő program/ a standard, MDS-8ØØ-ra vonatkozó verzió. Ezt a memóriaképet ki kell tehát menteni egy file-ba, hogy később be lehessen építeni a saját CBIOS és CBOOT programot:

```
SAVE 34 CPMbb.COM↵
```

A MOVCPM által létrehozott memóriaképben a cimek az eredetihez képest el vannak tolva, mivel ekkor a TPA-ban helyezkedik el a rendszer, és itt kell futnia. A memóriaképet a DDT-be töltve, azt meg lehet változtatni a szükséges mértékben:

```
DDT CPMbb.COM↵
```

A DDT válasza:

```
NEXT  PC
23ØØ  Ø1ØØ
-
```

A DDT listázó és visszafejtő parancsát használhatjuk ezután a 9ØØH és 227FH közti cimek vizsgálatára. A cimek kiszámitásához szükséges egy n negativ cim-eltolás értékének meghatározása. A 2Ø K-s rendszerben pl. a CCP a 34ØØH cimen kezdődik, a memóriaképben viszont a 98ØH cimen. Emiatt az n negativ cimeltolás értékére:

n+34ØØH=98ØH azaz n=98ØH-34ØØH=D58ØH

a kettes komplemens kivonás szabályai szerint.

2ØK-nál nagyobb rendszerek esetében:

n=D58ØH-b

Néhány általános memóriaméret esetében ez az érték:

24K	n=C58ØH
32K	n=A58ØH
48K	n=658ØH
64K	n=258ØH

A memóriaképben levő cim kiszámitásához tehát a CP/M cimhez hozzá kell adni az n cimeltolás értékét. Ez legegyszerübben a DDT program H parancsával hajtható végre, amely kiszámitja megadott két hexadecimális szám összegét és különbségét, ezek közül ez esetben az előbbire van szükség.

A BIOS cime a 2ØK-s rendszerben 4AØØ, a memóriaképben tehát 4AØØH+D58ØH=1F8ØH. Az L 1F8Ø parancs segitségével tehát a BIOS assembly kódját kiirathatjuk a képernyőre.

Ezután következhet a CBOOT és a CBIOS program elhelyezése a rendszerben. A BOOT /kezdeti betöltő/ program a memóriaképben a 9ØØH cimen kezdődik a valódi CP/M-ben viszont a 8ØH cimen. A cimeltolás

értéke tehát a BOOT esetében pozitiv, értéke pedig m=9ØØH-8ØH=88ØH. A DDT segitségével a CBOOT az alábbiak szerint tölthető be:

-I CBOOT.HEX↵

-R m↵

a második parancs m értékü cimeltolással /88ØH/ olvassa be a programot a memóriába. A

-L 9ØØ↵

begépelésével a DDT visszafejtő /disassembler/ programja inditható, mellyel megvizsgálható a CBOOT program.

Hasonlóképp olvasható be a CBIOS program is:

-I CBIOS.HEX↵

-R D58Ø↵

A második parancsban levő cimeltolás 2ØK-s rendszer esetén érvényes. A betöltés helyessége ismét az L paranccsal ellenőrizhető:

-L 1F8Ø↵

Ha meggyőződtünk a CBOOT és a CBIOS helyes betöltéséről, CTRL-C vagy GØ kiadásával visszatérhetünk a rendszerbe.

Ezek után már csak ki kell menteni az uj rendszert a lemezre a SYSGEN parancs segitségével:

```
SYSGEN↵

SYSGEN VERSION m.n
SOURCE DRIVE NAME /OR RETURN TO SKIP/ ↵
DESTINATION DRIVE NAME /OR RETURN TO REBOOT/ B↵
DESTINATION ON B THEN TYPE RETURN ↵
FUNCTION COMPLETE
DESTINATION DRIVE NAME /OR RETURN TO REBOOT/
```

Akimentett rendszer ekkor a B lemezen van, ezt áttéve az A lemezegységbe és hideg inditást kezdeményezve, betölthető az ujonnan konfigurált CP/M rendszer.

F Ü G G E L É K E K

I. FÜGGELÉK: CP/M PARANCSOK

Az alábbiakban tömör formában összegyüjtve közöljük a CP/M parancsok listáját. Az egyes parancsokat aszerint csoportositva adjuk meg, hogy perifériákkal, file-okkal, vagy programokkal kapcsolatos-e a funkciójuk; ezen belül ábécé sorrendben közöljük a parancsok formáját és müködésük tömör leirását. A parancsok használatához szükséges részletesebb leirás a kézikönyv megfelelő fejezeteiben található, a függeléket csak emlékezetfrissitőnek szántuk.

CP/M parancsok

Perifériakezeléssel kapcsolatos parancsok:

d:↵	a kurrens lemezegység megváltoztatása; d az uj kurrens egység szimbóluma /A, B, C, D/;
MOVCPM	a CP/M rendszer más memóriaméret kezeléséhez alkalmas, az aktuálistól eltérő verzióját hozza létre a MOVCPM program aktivizálásával;
PIP↵	aktivizálja a periféria-kezelő programot /Peripheral Interchange Program/;
PIP cél=forrás↵	másolás egyik lemezről egy másikra;

PIP paraméterek — különböző perifériaegységeken nyomtatási, kiirási, másolási, összemásolási és egyéb filemüveleteket végez /lásd a file-okkal kapcsolatos parancsoknál, később/;

STAT ↵ — képernyőre irja az aktuális perifériakiosztást;

STAT periféria-egymáshozrendelés↵ — megváltoztatja az aktuális logikai--fizikai periféria-egymáshozrendelést

Megjegyzés: a STAT VAL: parancs közli a képernyőn a lehetséges egymáshozrendeléseket;

STAT d:↵ — közli a képernyőn a kurrens lemezegységben lévő lemezen/vagy alternative a d:-vel megadott lemezegységben lévő lemezen/ rendelkezésre álló szabad hely méretét;

USER n↵ — megváltoztatja a felhasználói területet; n* egy felhasználói terület száma. n nélkül a USER parancs hatására a képernyőn megjelenik a jelenlegi felhasználói terület száma. Csak a CP/M 2.2 verzióban és a későbbi verziókban használható.

File-okkal kapcsolatos parancsok:

DIR { egyedi file-név / csoportos file-hivatkozás } ↵ kiirja az adott lemezegységen lévő hivatkozott file-/oka/t kiterjesztésükkel együtt a képernyőre; ha paraméter nélkül áll, a kurrens egységen lévő összes file nevét listázza;

ED file-név ↵ a CP/M rendszer szerkesztő-programját aktivizálja. A program text--file-okat /szöveg-file-okat/ hoz létre, ill. már létező text-file-ok aktualizálására alkalmazható. A text-file assembly-nyelvü program is lehet;

ERA { egyedi file-hivatkozás / csoportos file-hivatkozás } ↵ törli a megadott file--t /vagy file-okat/;

PIP uj file-név=régi file-név ↵

másolatot készit egy file-ról és névvel látja el;

PIP d:uj file-név=d:régi file-név ↵

másolatot készit egy file-ról egy másik lemezre; d: a lemezegységeket azonositja. A lehetséges rövidítésekre vonatkozóan lásd a parancs részletes ismertetésével foglalkozó fejezetet;

PIP↵
* uj file-név=régi file-név↵
*d:uj file-név=d:régi file-név↵
.
.
.
*↵

számos egymást követő másolási müvelet elvégzéséhez alkalmazható. A PIP programból való visszatéréshez nyomjuk le a "kocsi-vissza" billentyüt;

REN uj file-név=régi file-név↵

átnevez egy file-t /megváltoztatja a file nevét/;

STAT { d: egyedi file-hivatkozás
       d: csoportos file-hivatkozás } ↵

közli a megadott file/ok/ méretét; a 2.2 vagy annál magasabb CP/M-verzióknál közli a file-ok egyéb jellemzőit is, mint pl. a csak olvasásra megnyitható file-oknál azt, hogy a file R/O, vagy a rendszer-file-oknál azt, hogy a file un. SYS-file;

TYPE file-név↵ képernyőre irja egy text-file /szöveg-file/ tartalmát. Ha azt akarjuk, hogy a lista a sornyomtatón is megjelenjen, adjunk CTRL-P-t.

Programokkal kapcsolatos CP/M parancsok:

ASM file-név↵ assembly-nyelvü programokból végrehajtható programot készit;

DDT file-név↵ a CP/M debugger /hibafelderitő/ program segitségével felderiti egy program hibáit;

DUMP file-név↵ hexadecimális formában nyomtat ki egy végrehajtható formában levő programot tartalmazó file-t;

LOAD file-név↵ egy végrehajtható, tárgykódu programot tartalmazó file-nak .COM kiterjesztést ad, ezzel az adott programot végrehajtható uj külső parancsnak deklarálja;

Programnév↵ végrehajt egy tranziens parancsot vagy futtat egy programot. A programnév egy .COM kiterjesztéssel rendelkező file neve a ".COM" nélkül;

SAVE p file-név↵ DDT-vel történt betöltés után megőrzi /lemezre helyezi/ egy külső parancs, ill. program egy másolatát. A p az un. "lapok" száma. Egy "lap" 256 byte-nak felel meg;

SUBMIT file-név par1 par2 ...↵

elindit egy un. "batch"-feldolgozást. A file-név annak a file-nak a neve, melyben egymás után szerepelnek a végrehajtandó parancsok /programok/. A par1 par2, stb. az egyes parancsoknak /programoknak/ átadandó paramétereket jelölik. A parancs hatására a CP/M megkeresi a megadott file-t, behelyettesiti a paramétereket, és sorban végrehajtja a parancsokat. A feldolgozás a RUBOUT használatával bármikor megszakitható. Ha a file-ban szerepel az XSUB parancs, a végrehajtandó programok inputját is a parancs--file tartalmazza.

II. FÜGGELÉK: AZ EDITOR UTASITÁSAI

nA — n sort /vagy, ha n-et nem adjuk meg, 1 sort/ másol a forrás-file-ból az editor pufferének végére. Ha az n helyét "#"-t adunk meg, 65535 sort fog a pufferbe átvinni /feltölti a puffert/, ha Ø-t, félig fogja feltölteni a puffert /az átvitt sorok száma függ a puffer méretétől és a használatban lévő editortól/.

+/-nB — a karakter-mutatót /a továbbiakban CP-character pointer / a puffer elejére /+B/ ill. végére /-B/ pozicionálja.

+/-nC — A CP-t n karakterpozicióval előre /+/ vagy hátra /-/ mozgatja. A kocsi-vissza /CR/ ebből a szempontból két karakternek számit /egy ujsor és egy soremelés/.

+/-nD — + esetén azzal a karakterrel kezdődően, amelyre a CP mutat, töröl n sort, - esetén törli a CP-vel mutatott karakter előtti n-számu karaktert. Ha nem adjuk meg n-et, csak a CP által mutatott karaktert törli.

E — lezárja rendben a szerkesztési menetet. Az E parancs a pufferben lévő anyagot és a forrás--anyag még feldolgozatlan részét egy átmeneti output-file-ba teszi, majd ezt átnevezi arra a névre, amilyen neve a forrás-file-nak volt /ezzel párhuzamosan a forrás-file-t egy .BAK--tipusu file-ba menti, hogy megmaradjon az

eredeti forrásfile is/. A szerkesztőmenet véget ér, visszajön a rendszer.

nF jelsorozat { CTRL-Z / ↵ }

megkeresi a megadott jelsorozat n-számu előfordulását /ha n-et nem adjuk meg, n=1/ a CP-vel mutatott karaktertől kezdődően a pufferben előre haladva, majd a keresés végeztével a CP a megkeresett jelsorozat végére pozicionálódik. Ha a sorban még további parancsokat akarunk adni az editornak, a jelsorozatot CTRL-Z-vel kell lezárnunk, ha nem, a kocsi-vissza /CR/ billentyü lenyomásával kell lezárnunk a megadott jelsorozatot.

H lezárja a folyamatban lévő szerkesztőmenetet, végrehajt egy E parancsot, ezután ujra behivja az editort a most már uj forrás-file-ra /hatásában megfelel az eddigi szerkesztések mentésének és az editorhoz való visszatérésnek/.

I a CP mögé beszur egy uj sort; a CP-t az utoljára beszurt sor mögötti pozicióra állitja. A CP/M 2.2 verziójában, ha I-t adunk, a beszurt szövegben mindenhol a klaviatura billentyüin felül szereplő jelek kerülnek be a beszurt szövegbe a billentyük lenyomásakor, ha i-t adunk meg, annak megfelelő jelek kerülnek a beszurt szövegbe, hogy

használtuk-e a billentyü lenyomásával egyidejüleg a váltó /shift/ billentyüt is.

I szöveg CTRL-Z a CP mögé karaktereket szur be és a CP-t az utoljára beszurt karakter mögé állitja.

nJ jelsorozat1 CTRL-Zjelsorozat2 CTRL-Zjelsorozat3 { CTRL-Z / ↵ }

szöveg-jelsorozatokat helyez egymás mellé: megkeresi jelsorozat1-et, közvetlenül mögé elhelyezi jelsorozat2-t, törli a jelsorozat1 és jelsorozat3 közötti eddigi karaktereket és CP-t jelsorozat3 első karakterére állitja.

+/-nK + esetén törli a következő n-számu sort /ebbe beleértendő az a sor is, amelyen belülre a CP mutat; a sornak csak a CP-vel mutatott karakterrel kezdődő része törlődik/; - esetén törli a megelőző n-számu sort, melybe bele kell érteni azt a sort is, amelyen belülre a CP mutat; a sornak csak a CP-vel mutatott karakter előtti része fog törlődni.

+/-nL ha n=∅, a CP-t az aktuális sor elejére állitja; ha nem, ezen kivül n-sorral előre /+/ illetve hátra /-/ is állitja CP-t.

nMjelsorozat { CTRL-Z / ↵ }

Ha n nagyobb egynél, a jelsorozatként megadott ED-parancsokat n-szer hajtja végre.

Ha n nulla vagy egy, az M-parancs mindaddig ismétli a megadott ED-parancsok végrehajtását, mig hiba nem következik be.

nNszöveg { CTRL-Z / ↵ }

megkeresi a pufferben /és ha az nem elég, a még hátralévő forrás-anyagban/ a megadott szöveg n-számu előfordulását, és a CP-t a megtalált szöveg mögé pozicionálja. Mindaddig másol a forrás-file-ból a pufferbe, mig szükséges.

O abbahagyja a folyamatban lévő szerkesztőmenetet és visszatér az eredeti forrás-filehoz.

+/-nP A CP pozicionálása mellett a pufferben lévő szövegből oldalakat ir a képernyőre és a sornyomtatóra /oldalanként 22 sort/. n a nyomtatandó oldalak száma. +n a CP-t követően következő n-számu oldalt nyomtat, mig -n a CP előtti n-számu oldalt irja ki. Ha n-nek nulla értéket adunk, az aktuális sor és az azt követő 21 sor kerül kiirásra. A CP az utolsó kinyomtatott oldal első poziciójára mutat.

Q abbahagyja az aktuális szerkesztést, érintetlenül hagyva az átmeneti file-t, a forrás-file-t és a puffer-t, és visszatérést eredményez a CP/M-hez. A forrás-file

.BAK-tipusu biztonsági másolata nem készül el, de ha korábbról már volt ilyen azonos névvel, ezt törli /erre legyünk tekintettel!/.

R az X$$$$$$$$.LIB file tartalmát a CP mögé másolja és a CP-t a beszurt utolsó sor mögé állitja. A .LIB-file tartalma változatlan marad.

Rfilė-név a megadott nevü, .LIB-tipusu file tartalmát a CP mögé másolja és a CP-t az utolsó beszurt sor mögé állitja. A .LIB-file nem változik.

nSrégi-szöveg CTRL-Zuj-szöveg {CTRL-Z / ↵}

a CP mögötti puffertartalomban megkeresi a "régi-szöveg"-ben megadott szövegtöredéket, és helyére az "ujszöveg"-ben megadott szöveget iktatja, mindezt n-szer, ha n nagyobb egynél.

+/-nT ha n-t nem adjuk meg vagy n=1, a CP utáni karaktertől az aktuális sor végéig jelenik meg a sor tartalma a képernyőn; ha n=∅, a képernyőn kiirásra kerülnek az aktuális sor CP-t megelőző karakterei. Ha n pozitiv /+/, az aktuális sorral kezdődően n-számu sort ir ki a képernyőre. Ha n negativ /-/, az aktuális sort megelőző n-számu sor kerül a képernyőre plusz az aktuális sornak a

CP-t megelőző része. A B#T parancsok hatására a teljes puffer kiirásra kerül.

+/-U — +U hatására minden ezután bevitt vagy beszurt karakter helyett a megfelelő felső váltó/shift/-állásbeli jel kerül be a szövegbe. Az átalakitást -U kiadásával oldhatjuk fel.

V — hatására a képernyőn sorszámmal ellátva jelennek meg a puffer sorai /a file-ban valójában nincsenek megszámozva a sorok/.

ØV — hatására kiiródik a képernyőre a puffer-beli szabad hely byte-okban és a puffer teljes mérete /mindkettő decimális számok formájában/. Például, a "27648/28832" válasz esetén 27648 byte-nyi szabad hely van a 28832 byte méretü pufferban.

nW — a .$$$ file-név kiterjesztéssel rendelkező átmeneti output file-ba kiirásra kerül az aktuális és az azt követő n-számu sor. Ha nem adjuk meg n-et, csak az aktuális sor kerül kiirásra.

nX — a következő n-számu sort kimásolja az X$$$$$$$.LIB nevü file-ba /a másolt sorok a szerkesztés alatt álló anyagban változatlanok maradnak eközben, tehát nem törlődnek/. A kivitt sorok visszanyerését az R parancs

teszi lehetővé. Ha n=∅, a parancs hatására törlődik az X$$$$$$$$.LIB file.

**nZ — n óraütés időtartamra felfüggeszti az ED programot /ez megfelel kb n másodpercnek/.

+/-n — megfelel egy "+/-nLT" parancs-kombinációnak.

n: — a CP-t az n-edik sor elejére pozicionálja.

n1::n2 — az n1 sorral kezdődő és az n2 sorral végződő szövegrészt specifikál. Ha valamelyik a kettő közül hiányzik, ez ugy tekinthető, mintha a helyén az aktuális sor állna.

III. FÜGGELÉK: LOGIKAI ÉS FIZIKAI ESZKÖZNEVEK

LOGIKAI ESZKÖZNEVEK

CON: a "konzolt" vagy terminált jelöli, mely klaviaturával és képernyővel is rendelkezik /input-output/

RDR: papirszalag vagy kártyaolvasó /csak input/

PUN: papirszalag vagy kártyalyukasztó /csak output/

LST: egy "listázásra" alkalmas egységet jelöl, mint amilyen pl. egy sornyomtató /csak output/

FIZIKAI ESZKÖZNEVEK

TTY: jelölhet konzolt/terminált/, olvasót, lyukasztót vagy lista-perifériát

CRT: lehet konzol /terminál/, vagy lista-periféria /Cathode Ray Tube - katódsugárcső/,

PTR: papirszalag vagy kártyaolvasó egység

LPT: listázásra használt eszköz /sornyomtató/

UC1: felhasználó által definiált konzol /terminál/

UR2: a második, felhasználó által definiált olvasó

UR1: felhasználó által definiált olvasó

UR2: a második felhasználó által definiált olvasó

UP1: felhasználó által defin9ált output /"lyukasztó"/-eszköz

UP2: a második, felhasználó által definiált output-/"lyukasztó"/-eszköz

UL1: felhasználó által definiált kiiró egység

Megjegyzés: A BAT: azért nem szerepel a felsorolásban, mert az pusztán az RDR: és LST: értékek ujrakiosztását jelenti.

IV. FÜGGELÉK: PIP KULCSSZAVAK ÉS PARAMÉTEREK

PIP KULCSSZAVAK

NUL: 4Ø db nullát /ASCII Ø-kódot/ küld az egységre, mely rendszerint egy "lyukasztó" /output/ egység. Ennek példája, amikor PROG.HEX-et küldjük a lyukasztóra:

```
*PUN:=PROG.HEX,NULL:
```

EOF: file-vége /ASCII CTRL-Z-kód/ kódot küld az egységre /ezt egyébként a PIP az ASCII-ben történő file-mozgatásoknál automatikusan megcsinálja, igy explicite csak speciális esetekben szükséges az alkalmazása/. Példa:

```
*PUN:=NUL:,X.ASM,EOF:,NULL:
```

A példában 4Ø db nulla megy a lyukasztó-egységre, ezt követi az X.ASM-file másolata, melyet a file-vége karakter és további 4Ø db Ø követ.

PRN: megegyezik az LST:-vel azzal a különbséggel, hogy nyolc karakterenként tabulál, a sorokat számozza /hasonlóan az ED programhoz/, 60-soronként lapot emel,akárcsak a nyomtatás megkezdése előtt. Példa:

```
*PRN:=SAMPLE.TXT
```

INP: speciális input-eszköz kezelő rutin, amely beilleszthető magába a PIP-programba /a rutint nekünk kell megirnunk assembly-nyelven, majd a PIP-et ezzel ki kell egészitenünk/. A PIP az inputot karakterről karakterre fogadja egy konkrét memóriacimre

vonatkozó call végrehajtásával /1Ø3H/, és az adatokat a 1Ø9H cimtől kezdődően tárolja /a paritás--bitnek Ø-nak kell lennie/

OUT: speciális output-eszköz kezelő rutin, amelyet beilleszthetünk magába a PIP-be, mint az INP:-nél láttuk. A PIP a 1Ø6H cimre ad ki call-t és az adatot a C-regiszterbe küldi /karakterenként/. Az assembly--nyelvü programozóknak szóló megjegyzés: a 1Ø9H-1FFH intervallumba eső memóriacimeket a PIP nem használja, igy oda speciális célu periféria driver-rutinok helyezhetők el/a DDT-t segitségével/. Példák:

*GIZMO.CLK=INP:

/a speciális egységről érkező inputot a GIZMO.CLK file tartalmazza/;

*OUT:=GIZMO.CLK

/a GIZMO.CLK tartalma kikerül a speciális output perifériára/.

PIP PARAMÉTEREK

[B] blokk-módu átvitel. A PIP az adatokat egy pufferbe helyezi mindaddig, mig egy ASCII CTRL-S karaktert nem olvas az eszközről. Ezután kiüriti a lemez-puffert és folytatja az adatok olvasását. A puffer mérete függ a futtató CP/M-rendszer méretétől /lásd a rendszerrel együtt szállitott dokumentációt/. Olyankor célszerü ezt a paramétert használni, ha egy folyamatosan olvasó eszközről, mint pl. a kazettás egység, történik az átvitel.

Példa:

*ENUFF.TXT=RDR: [B]

[Dn] a PIP törli az n. oszlopon tulnyuló karaktereket /a képernyőn kiirt oszlopról van itt szó/ text -szöveg-file-ok másolása közben. Hosszu sorok csonkolására használhatjuk olyankor, amikor "keskeny" perifériára küldünk ki file-okat. Példa:

*PRN:=LONG.TXT [D52]

[E] A végrehajtás közben minden másolási müvelet jelenjen meg a képernyőn is /echo - visszhang/. Példa:

*CPPY.TXT=SOURCE.TXT,S2.TXT,S3.TXT,S4.TXT[E]

[F] a PIP kiszüri /azaz eltávolitja/ a lapemeléseket a file-ból. A[P]paraméter alkalmazásával ujra lehetséges a lapemelések beiktatása.

[Gn] az n-edik user területről veszi a file-t /CP/M 2.2 verziótól felfelé alkalmazható/.

[H] hexadecimális adatátvitel: a PIP minden adatot ellenőriz, hogy szabványos Intel hexadecimális formátumu-e.

[I] az Intel hexadecimális formátumu file-ok átvitelénél elhanyagolja a ":ØØ" rekordokat /automatikusan beállitja a H paramétert/.

[L] minden felső váltó - shift állásnak megfelelő karaktert konvertál a megfelelő alsó váltó-állásbeli karakterre.

[N] az uj file-ba másolt minden sort megsorszámoz /a sorszámozás 1-től kezdődik/. Minden sorszámot kettőspont követ. A vezető nullák /pl. ØØ3/ nem kerülnek kiirásra, hacsak nem az N2 paramétert használjuk, mely esetben a vezető nullák megmaradnak és a sorszám után tabulálás következik. A tabulálásra egyébként a T paraméter használható.

[O] tárgykódu file átvitele /nem ASCII file-ok/: a PIP figyelmen kivül hagyja az összefüzés közben a logikai file-véget /CTRL-Z-t/.

[Pn] A PIP minden n-edik sor után, akárcsak az átvitel megkezdése előtt, lapot emel. Ha n=1, vagy nem adjuk meg, a lapemelés 6Ø soronként következik be. Ha használjuk az F paramétert is, a PIP először kiiktatja az inputon lévő lapemeléseket és ezután lép életbe az output-on a lapemelés.

[Qjelsorozat CTRL-Z]

a PIP a megadott jelsorozat megtalálásáig folytatja csak az átvitelt. A jelsorozatot CTRL-Z zárja le. Részletesebben lásd a file-ok részleges másolásáról szóló részt.

[R] rendszer /$SYS/ file-okat olvas /másol/; a W paraméternek megfelelő müveletet is végrehajtja /csak a CP/M 2.2 verziótól felfele alkalmazható/.

[Sjelsorozat CTRL-Z]

A PIP az egységről vagy file-ból történő másolást akkor kezdi el, ha megtalálta a specifikált

jelsorozatot. A megadott jelsorozatot CTRL-Z-vel kell lezárni. Részletesebben lásd a file-ok részleges másolásáról szóló részt.

[Tn] text - szöveg file-ok átvitele közben minden n--edik oszlop után tabulál. Egy text - szöveg file-ban a CTRL-I használatával érhetünk el tabulálást; ezzel a paraméterrel tetszőleges helyen tabulálhatunk, nemcsak az általában kötött helyen lévő oszlopoknál.

[U] text - szöveg file-ok átvitele közben minden alsó váltó - shift állásnak megfelelő karaktert ugyanazon billentyü felső váltó - shift állásbeli megfelelő karakterévé konvertál át.

[V] a PIP ellenőrzi, hogy az adatok átvitelénél nem történt-e átviteli hiba: az átvitel után ujraolvassa a frissen elkészült output-ot /az output tehát nem lehet periféria, lemezes file-nek kell lennie/ és sikeres másolás esetén ilyen értelmü üzenetet küld a képernyőre.

[W] felülirja /törli a csak olvasásra megnyitható fileokat /figyelmen kivül hagyja a $R/O jelzést/. Csak a CP/M 2.2 verziótól felfelé használható.

[Z] ASCII karakterek inputja esetén nullára állitja a paritás-bitet. Különösen olyankor használjuk ezt a paramétert, ha speciális input-egységről /:INP/ történik a beolvasás.

V. FÜGGELÉK: KÉPERNYŐSZERKESZTÉS: VEZÉRLŐ /KONTROL/ KARAKTEREK

ÁLTALÁNOS VEZÉRLÉS

rubout/delete	utolsó karakter törlése és visszajelzése
CTRL-U vagy CTRL-X	sor törlése
CTRL-R	sor ujrairása
CTRL-E	folytatás a következő sorban
CTRL-C	a CP/M ujrainditása

EGYÉB

CTRL-H	visszaléptetés
CTRL-I	8-karakterenkénti tabulálás
CTRL-J /soremelés/	input lezárása
CTRL-M/kocsivissza/	parancs befejezése
CTRL-P	sornyomtató ki-bekapcsolás
CTRL-S	konzol-output leállitása-ujrainditása
CTRL-Z	konzol-input befejezése /PIP és ED/

VI.FÜGGELÉK: A CP/M FONTOSABB HIBAÜZENETEI

A rendszer használata során háromféle hiba szokott előfordulni. Mindhárom esetben ugyanolyan általános formáju üzenetet kapunk:

BDOS ERR ON d: hiba

ahol d jelzi, hogy melyik lemezegységen történt a hiba, mig "hiba" az alábbi hibaüzenetek egyike:

BAD SECTOR	/rossz szektor/
SELECT	/kiválasztás/
READ ONLY	/csak olvasás/

A CP/M 2.2 verziójában van még egy negyedik hiba-eset is, amikoris a hibaüzenet:

FILE R/O

BAD SECTOR:

A "bad sector" - rossz szektor olyankor jelentkezik, amikor a lemez-vezérlő nem képes a lemezegységről információkat visszanyerni. Ez olyankor fordulhat elő, ha a lemez maga sérült /kopott/ - fizikai hibás szektor van rajta, vagy ha a lemezvezérlő müködésében áll be zavar. A hiba másik oka az lehet, hogy a lemezegységben nem volt bent a kivánt lemez, amikor kisérlet történt a lemezről való olvasásra. Olyankor is ezzel a hibával találjuk szembe magunkat, ha a jelenleg használttól eltérő lemezvezérlő alkalmazása mellett történt a lemezen az információ

rögzitése. Még olyankor is előfordulhatnak apró eltérések a rekordformátumok között, ha mindkét lemezvezérlőről azt állitja a forgalmazó cég, hogy azok "IBM-kompatibilisek". Pl. az Intel MDS-800 vezérlő használata mellett floppy-lemezre irt file-okat más vezérlők nagy valószinüséggel el fogják tudni olvasni, mig más vezérlőkkel a lemezre rögzitett információknak MDS-800-as vezérlővel való olvasásakor könnyen előfordulhat a BAD SECTOR hiba. Hasonlóképpen, ha egy file-ban lévő információ a lemez nem megfelelő kezelése következtében fizikailag, vagy hibás programmüködés következtében logikailag sérült, ugyanezt a hibaüzenetet kapjuk.

A hiba előfordulása után két lehetőségünk van:
vagy a CTRL-C lenyomásával ujrainditjuk a rendszert, ezzel természetesen félbehagyva azt a feldolgozást vagy programfutást, amely közben a hiba felmerült, vagy figyelmen kivül hagyjuk a hibát és folytatjuk a program futtatását, ill. a file feldolgozását a CR-"kocsi-vissza" billentyü /RETURN/ lenyomásával, melynek hatására a rendszer figyelmen kivül hagyja a hibás szektort.

Nem tulzottan tanácsos az ilyen hibát figyelmen kivül hagyni! Ha a program, vagy a file-müvelet használja éppen a lemez directory-ját, a hiba figyelmen kivül hagyásának következménye igen gyakran az, hogy a lemez elvesziti integritását, és használhatatlanná válik. Mindig ügyeljünk arra, hogy megfelelő biztonsági másolatokkal rendelkezzünk minden lemezünkről.

SELECT: Ez a hiba olyankor lép fel, ha egy olyan lemezegységet választunk ki, amilyen nem létezik. A d-vel megadott érték jelenti az általunk kiválasztott, és hibásan kiválasztott egységet. A rendszer automatikusan ujrainditja magát a hiba előfordulása után az első, tetszőlegesen választott billentyü lenyomása hatására.

READ ONLY: Ez a hiba akkor jelentkezik, ha megpróbálunk irni egy olyan lemezre, melyet a STAT használatával /vagy egy olyan program által, amely egy BDOS funkciót használ/ "csak olvasás"-ra használhatóként jelöltünk meg. Olyankor is bekövetkezhet, amikor uj lemezt helyezünk el a rendszerben anélkül, hogy CTRL-C-vel ujraindítottuk volna rendszerünket; minden uj lemez behelyezésekor CTRL-C-t kell végrehajtanunk, hogy az uj lemezre irni tudjunk /file-ok felülirása, file-ok törlése, file-ok létrehozása, file-ok tartalmának aktualizálása/.

Tetszőleges billentyü lenyomásával szüntethetjük meg a hibát; a rendszer ilyenkor automatikusan ujrainditja magát, melynek hatására a lemez irásra és olvasásra egyaránt használható lemezként kerül be a rendszer lemeznyilvántartásába.

FILE R/O: Ez a hiba a CP/M-nek csak az ujabb verzióinál léphet fel /CP/M 2.2 és későbbi verziók/ olyankor, ha megpróbálunk irni /felülirni,

aktualizálni vagy törölni/ egy olyan file-t, amely az $R/O tulajdonsággal /csak olvasásra/ rendelkezik /ezt a jellemzőt vagy a STAT parancs, vagy egy felhasználói program adhatja egy file-nak/.

Ezt a hibát ugy oldhatjuk fel, hogy izlés szerint valamelyik billentyüt lenyomjuk a klaviaturán. A csak olvasásra megnyitható file-lal kapcsolatos müvelet ennek hatására félbeszakad, és ezután meg kell változtatnunk /$R/O-ról $R/W-re a file megfelelő tulajdonságkódját, hogy a file-ra irni tudjunk. A file-jellemzők módositására a STAT parancsot alkalmazhatjuk.

VII. FÜGGELÉK: A CP/M ÁLTAL TÁMOGATOTT FILE-TIPUSOK /file-név kiterjesztések/

.ASM	Assembly forrásnyelvü programok
.BAK	az ED/editor - szerkesztő/ program által generált biztonsági forrás-file másolat
.BAS	BASIC-E, CBASIC vagy MBASIC forrásnyelvü programok
.CRF	relokálható /áthelyezhető/ assembler kereszthivatkozás
.COB	COBOL-forrásnyelvü programok
.COM	végrehajtható programok /tranziens parancsok/ file-ja
.DAT	ASCII adatfile
.FOR	FORTRAN forrásprogram
.HEX	leforditott /assemblált/ assembly-nyelvü programok
.INT	BASIC-E, CBASIC tárgy-kódu programok
.MAC	relokálható /áthelyezhető/ assembler forrás--kód
.LST	BASIC-E, CBASIC programlista
.OVR	COBOL-fordító overlay
.PRN	lista /assembler, FORTRAN, COBOL, stb./
.REL	áthelyezhető tárgykódu program /relokálható assembler, FORTRAN, stb./

.SUB	SUBMIT parancs-file
.SYM	assembler szimbólum-táblázat
.XRF	abszolut assembler kereszt-hivatkozási táblázat
.$$$	ED vagy PIP átmeneti file-ja

VIII. FÜGGELÉK: CP/M - KOMPATIBILIS PROGRAMOK

A CP/M operációs rendszer alá igen sok olyan program készült, amelyek megkönnyitik a rendszer használatát. Számos szövegfeldolgozó, BASIC és magasszintü nyelv is fut a CP/M alatt. Az alábbiakban ezek közül ismertetünk néhányat röviden.

Utility programok

COPY — Lemezmásolat készitése. Minden track-et átmásol.

DUMP — Lemez tartalmának kiiratása. A kiiratás történhet file-onként, sávonként, szektoronként vagy egy file-hoz allokált szektorok szerint. Használható véletlenül törölt vagy más módon elvesztett file-ok visszaállitására.

XDIR — A directory alfabetikus sorrendben történő kiiratása. Kiirja a lemezen maradt szabad terület méretét is.

XREF — Az ASM-mel előállitott listához kereszthivatkozási /crossreference/ lista készitése.

COMPARE — Két file összehasonlitása, az eltérések kiiratása.

ERASE Biztonságosabb törlés, megkérdezi a felhasználót, hogy valóban törölni akarja-e a file-t. y /yes-igen/ válasz esetén törli. n/no/ válasz esetén nem törli.

UNERASE Véletlenül törölt file visszaállitása. Csak az a file állitható vissza, amelyhez allokált blokkok /másnéven file-területek/ nem lettek időközben más file-hoz allokálva.

FIND Egy megadott filenév-csoporthoz illeszkedő fileneveket keresi meg a definiált lemezeken.

SPACE Megadja, hogy az egyes felhasználói területek mekkora lemezterületet foglalnak le, és menynyi a teljes lefoglalt, illetve szabad terület a lemezen.

MOVE File másolása egyik felhasználói területről a másikra ugyanazon a lemezen. Csak a directory-ban változtatja meg a file-hoz tartozó felhasználói terület számát, ez jóval gyorsabb; mint a file fizikai átmásolása.

MAKE File-ok "láthatatlanná", illetve "láthatóvá" tétele. A láthatatlan file-ok a Ø-ás felhasználói területen vannak, rendszer státuszuk van /SYS attributum/ és csak olvashatóak. A látható file-ok irható/olvasható-ak, és directory státuszuak. Amikor láthatatlanná teszünk egy file-t a MAKE segitségével, az

átiródik a kurrens felhasználói területről a Ø-ás felhasználói területre. Láthatóvá tételkor ennek a forditottja történik.

SPEED — Soros vonalat használó perifériák /TERMINÁL, PRINTER, MODEM/ átviteli sebességének lekérdezése és beállitása. A beállitható sebességek: 3ØØ, 6ØØ, 12ØØ, 24ØØ, 48ØØ, 96ØØ és 192ØØ baud.

PROTOCOL — Soros vonalat használó eszközök átviteli protokolljának beállitása.

ASSIGN — Logikai eszköznevek hozzárendelése fizikai eszközökhöz /perifériákhoz/. Egy fizikai eszköz több logikai eszköznévhez is hozzárendelhető. Pl. A CONSOLE INPUT és a CONSOLE OUTPUT egyaránt lehet a TERMINAL nevü fizikai eszköz. A megengedett logikai eszköznevek: CONSOLE, AUXILIARY, LIST; a fizikai eszköznevek: TERMINAL, PRINTER, MODEM. Funkcióját tekintve hasonló a STAT parancshoz.

DATE — A napi dátum /év, hónap, nap/ megadása és egy flag beállitása, amely jelzi, hogy a dátum aktualizálása megtörtént.

TIME — Az időpont /óra, perc, másodperc/ megadása, és egy flag beállitása, amely jelzi, hogy az időpont aktualizálása megtörtént.

Szövegszerkesztők és szövegformázók

WORD-STAR — Teljes képernyőnyi anyagon egyszerre manipuláló szöveg-szerkesztő /screen-editor/ és szövegformázó.

TEX — Lapméret, margó, sorhosszuság, bekezdés, stb. beállitási lehetőséget biztositó szövegformázó.

TSC szövegfeldolgozó

Felhasználói makrók definiálását megengedő szövegfeldolgozó.

Assemblerek

MAC — Az Intel ASM-80 nevü macro assembleréhez hasonló assembler. /Az azonositók hossza legfeljebb 16 karakter/ . A SID /lásd később/ által használt szimbólum-táblát állit elő. Z8Ø programok forditására is képes.

SMAL/8Ø — Strukturált macro assembler, a magas szintü nyelvekhez /főként a C-hez/ hasonló szintaktikája van. Utasitásai között megtalálhatók az IF THEN ELSE, REPEAT, LOOP, WHILE.
A szubrutinok tetszőleges mélységig egymásba ágyazhatók, a rekurziv hivás megengedett.
A SMAL/8Ø programok sokkal "olvashatóbbak", mint az assembly szintü programok.

Z8Ø-ASM 8Ø8Ø-as processzoron futó Z8Ø keresztfordító cross-assembler.

MICROSOFT BASIC /MBASIC/

Hosszu változónevekeet megengedő, strukturált programozást támogató BASIC. Létezik fordítóprogram /compliler/ változata is.

CBASIC BCD-aritmetikát, WHILE ciklusutasitást, fejlett file-kezelést tartalmazó BASIC. Igen sok adatfeldolgozási programot irtak ezen a nyelven.

Forditóprogramok /compilerek/

FORTRAN-8Ø Az ANSI 66 szabványnak megfelelő /kivéve a COMPLEX adattipust/ FORTRAN fordító. Tartalmaz MACRO-8Ø assemblert, betöltő programot /Linking Loader/, könyvtárkezelőt és szubrutinkönyvtárat is.

PASCAL/M Az UCSD Pascal-lal kompatibilis, majdnem "teljes" Pascal; közbenső kódra /P-kód/ fordit, amelyet interpreter hajt végre.

PASCAL/Z Gépi kódra fordit, a Pascal egy részhalmazát /subset/ értelmezi.

Whitesmith C compiler

A Kernighan és Richie által definiált teljes C nyelv implementációja. Számos hibadiagnosztikai rutint tartalmaz, jelentős mértékü optimálást végez a fordítás során, amely 4 menetből áll /macro kifejtés, szintaktikus elemzés, közbenső kódra fordítás, a közbenső kód leforditása/.

BDS C compiler

A C nyelv egy részhalmazát kezelő fordító. Hiányoznak a hosszu adattipusok, a lebegőpontos számok, nincs adatinicializálás, STATIC és REGISTER deklaráció. A fordító optimálást sem végez. Tartalmaz viszont betöltőt és könyvtárkezelőt. Relokálható 8Ø8Ø-as kódot hoz létre.

Tiny-C

Gépfüggő interpreterből és gépfüggetlen programkészitő rendszerből /Program Preparation System, PPS/ áll. A PPS szövegszerkesztőt és a rendszer segitségével összeállitott programokat futtató monitort tartalmaz. Ez olyan kényelmes, interaktiv programfejlesztési lehetőséget biztosit, mint a BASIC interpreter. A program futtatása során, ha hibát észlel a futtató rendszer, félbeszakitja a végrehajtást és jól használható diagnosztikai üzenetet ad. A tiny-C-nek IF/ELSE, WHILE, RETURN és BREAK vezérlő utasitásai, egész és karakter tipusu

változói, egydimenziós tömbjei és olyan pointer konvenciója van, amely lehetővé tesz mind cim, mind érték szerinti paraméterátadást. A PPS lehetővé teszi tetszőleges számu programmodul létrehozását, ezeket a futtató monitor épiti össze. A modulok között lehetnek gépi kódu szubrutinok is.

Nyomkövetési segédeszközök

DDS-II

A DDS a következő értékeket irja ki a képernyőre a program futása közben:

- a stack pointer tartalma
- a stack tetején levő 5 szó
- a stack tetején levő 5 szó által megcimzett memóriahelyek tartalma
- a végrehajtásban következő öt utasitás assembly formája
- az AF, BC, DE, HL regiszterek és a PC /programszámláló/ regiszter tartalma a kurrens utasitás előtti állapotnak megfelelően
- a BC, DE, HL regiszterpárok által megcimzett memóriarekeszek tartalma
- a memória tetszőleges részének tartalma
- az utolsó öt visszatéritési cim, amely CALL utasitás eredményeként került a stack-be.

A DDS jelzi, ha a program olyan adaton manipulál, amely kivül esik egy előre definiált tartományon, vagy ha a program egy adott cimre hivatkozik, vagy ha az SP egy előre megadott értéket meghalad, vagy ha a program egy megadott cimre ir, vagy ha egy regiszter vagy memória byte vagy szó felvesz egy előre definiált értéket.
Ezeken kivül a DDS tartalmazza a debug monitorok szokásos lehetőségeit /kiiratás, módositás, töréspontozás, lépésenkénti végrehajtás, stb./.

SID

A DDT hircm további lehetőséggel továbbfejlesztett változata. Két utilityt /TRACE és HIST/ tartalmaz, valamint lehetővé teszi, hogy a cimekre a forráskódban lévő cimkékkel hivatkozzunk. A TRACE utility egy előre megadott töréspont előtti programfutást rögzit. A HIST utility arról ad tájékoztatást, hogy mely programrészletek futnak leggyakrabban. A tapasztalatok szerint a programok futási idejük 9Ø-95 %-át a kód 5-1Ø %-ának végrehajtásával töltik. Ennek az 5-1Ø %-nak a megkeresését teszi lehetővé a HIST, mely programrész optimálása azután nagyban meggyorsithatja a program futását.

IX. FÜGGELÉK: DDT MINTAPÉLDA

```
D100 17F

0100 01 BC 0F C3 3D 01 43 4F ....=.CO
0108 50 59 52 49 47 48 54 20 PYRIGHT
0110 28 43 29 20 31 39 38 30 (C) 1980
0118 2C 20 44 49 47 49 54 41 , DIGITA
0120 4C 20 52 45 53 45 41 52 L RESEAR
0128 43 48 20 20 20 20 20 20 CH
0130 44 44 54 20 56 45 52 53 DDT VERS
0138 20 32 2E 32 24 31 00 02  2.2$1..
0140 C5 C5 11 30 01 0E 09 CD ...0....
0148 05 00 C1 21 07 00 7E 3D ...!..~=
0150 90 57 1E 00 D5 21 00 02 .W...!..
0158 78 B1 CA 65 01 0B 7E 12 x..e..~.
0160 13 23 C3 58 01 D1 C1 E5 .#.X....
0168 62 78 B1 CA 87 01 0B 7B bx.....+
0170 E6 07 C2 7A 01 E3 7E 23 ...z..~#
0178 E3 6F 7D 17 6F D2 83 01 .o|.o...
```

Commodore 64-es gépen bekapcsolás után közvetlenül készült memória dump 1ØØH-tól 17FH-ig

```
-F100 17F 20
```

A memória adott területének feltöltése szóközzel /2ØH-val/.

```
-D100 17F

0100 20 20 20 20 20 20 20 20
0108 20 20 20 20 20 20 20 20
0110 20 20 20 20 20 20 20 20
0118 20 20 20 20 20 20 20 20
0120 20 20 20 20 20 20 20 20
0128 20 20 20 20 20 20 20 20
0130 20 20 20 20 20 20 20 20
0138 20 20 20 20 20 20 20 20
0140 20 20 20 20 20 20 20 20
0148 20 20 20 20 20 20 20 20
0150 20 20 20 20 20 20 20 20
0158 20 20 20 20 20 20 20 20
0160 20 20 20 20 20 20 20 20
0168 20 20 20 20 20 20 20 20
0170 20 20 20 20 20 20 20 20
0178 20 20 20 20 20 20 20 20
```

A memória képe a feltöltési parancs végrehajtása után.

```
-S100
0100 20 41
0101 20 42
0102 20 433
0103 20 4
0104 20 5
0105 20 46
0106 20 47
0107 20 48
0108 20
```

Adott memóriacimtől byte-onként történő feltöltés /S - substitute parancs/.

Befejezés a "." karakterrel.

```
-D100 10F
0100 41 42 03 04 05 46 47 48 AB...FGH
0108 20 20 20 20 20 20 20 20
```

A memória képe a parancs végrehajtása után.

```
-H100
0100  MOV A,B
0101  POP H
0102  PUSH B
0103  RET
0104
```

Intel 8Ø8Ø mnemonikokkal történő programozás a 1ØØH cimtől kezdődően.

A befejezés itt is a "." karaktertel.

```
-D100 10F
0100 78 E1 C5 C9 05 46 47 48 x....FGH
0108 20 20 20 20 20 20 20 20
```

Az "assemblálás" utáni memóriakép.

```
-L100 104
  0100  MOV  A,B
  0101  POP  H
  0102  PUSH B
  0103  RET
  0104  DCR  B
  0105
```

A memória listázása assembly-mnemonikok segitségével /Intel 8080/

A programfutás nyomkövetése /trace funkció/

```
-T

C0Z0M0E0I0 A=00 B=0000 D=0000 H=0000 S=0100 P=0100 MOV  A,B*0101
-T

C0Z0M0E0I0 A=00 B=0000 D=0000 H=0000 S=0100 P=0101 POP  H*0102
-T

C0Z0M0E0I0 A=00 B=0000 D=0000 H=E178 S=0102 P=0102 PUSH B*0103
-T

C0Z0M0E0I0 A=00 B=0000 D=0000 H=E178 S=0100 P=0103 RET  *0000
```

3 lépésnyi program nyomkövetése

```
-T3

C0Z0M0E0I0 A=00 B=0000 D=0000 H=E178 S=0102 P=0000 JMP AA03
C0Z0M0E0I0 A=00 B=0000 D=0000 H=E178 S=0102 P=AA03 JMP AB1D
C0Z0M0E0I0 A=00 B=0000 D=0000 H=E178 S=0102 P=AB1D LXI SP,0080*AB20
```

A CPU regisztereinek vizsgálata /összes/

```
-X
C0Z0M0E0I0 A=00 B=0000 D=0000 H=E178 S=0080 P=AB20 MVI  C,00
```

A DE regiszterpár tartalmának megváltoztatása
és a CPU regisztereinek ujbóli megjelenitése

```
-XD
D=0000 5678
-X
C0Z0M0E0I0 A=00 B=0000 D=5678 H=E178 S=0080 P=AB20 MVI  C,00
```

A zéró-bit megváltoztatása és a változtatás
ellenőrzése a CPU regiszterek megjelenitésével

```
-XZ
Z0 1
-X
C0Z1M0E0I0 A=00 B=0000 D=5678 H=E178 S=0080 P=AB20 MVI  C,00
```

X. FÜGGELÉK: BIOS UGRÓTÁBLA FELÉPITÉSE

```
ØØH       JMP BOOT;      Hideg start
 WBOOTE:
Ø3H       JMP WBOOT;     Meleg start
Ø6H       JMP CONST;     Konzol státusz vizsgálat
Ø9H       JMP CONIN;     Karakter beolvasás a konzolról
ØCH       JMP CONOUT;    Karakter kiirás a konzolra
ØFH       JMP LIST;      Karakter kiirása a listázó eszközre
12H       JMP PUNCH;     Karakter kiirása a lyukasztóra
15H       JMP READER;    Beolvasás a lyukszalag olvasóról
18H       JMP HOME;      Ø. trackre állitás a kiválasztott
                         lemezen
1BH       JMP SELDSK;    Drive kiválasztás
1EH       JMP SETTRK;    Track beállitás
21H       JMP SETSEC;    Szektor beállitás
24H       JMP SETDMA;    DMA cim beállitás
27H       JMP READ;      Kiválasztott szektor beolvasása
2AH       JMP WRITE;     Kiválasztott szektor kiirása a
                         lemezre
2DH       JMP LISTST;    Listázó eszköz státuszának beol-
                         vasása
3ØH       JMP SECTRAN;   Szektor konvertáló rutin
```

b - ugrótábla kezdőcime. 2Ø K-s CP/M rendszerben

b = 4AØØH

XI. FÜGGELÉK: DPT és DPB

2-lemezes rendszer /8 inch-es IBM--kompatibilis, egyszeres sürüségü floppy-k/ esetén

```
;DISZK PARAMÉTER TÁBLA FEJLÉC A DRIVE-HOZ
DP BASE:    DW  XLT
            DW  ØØØØH
            DW  ØØØØH
            DW  ØØØØH
            DW  DIRBUF
            DW  DPBLK          ;Diszk Parameter Blokk
            DW  CSVØ
            DW  ALVØ

;DISZK PARAMETER TÁBLA FEJLÉC AZ 1. DRIVE-HOZ
            DW  XLT
            DW  ØØØØH
            DW  ØØØØH
            DW  ØØØØH
            DW  DIRBUF
            DW  DPBLK
            DW  CSV1
            DW  ALV1

;LOGIKAI -FIZIKAI SZEKTORSZÁM TRANSZFORMÁLÁS
XLT:        DB  1, 7, 13, 19
            DB  25, 5, 11, 17
            DB  23, 3, 9, 15
            DB  21, 2, 8, 14
            DB  2Ø, 26, 6, 12
```

```
            DB   18, 24, 4, 1Ø
            DB   16, 22

;DISZK PARAMÉTER BLOKK; MINDKÉT DISZKHEZ UGYANAZ

DPBLK:      DW   26          ;SPT
            DB    3          ;BSH
            DB    7          ;BLM
            DB    Ø          ;EXM
            DW  242          ;DSM
            DW   63          ;DRM
            DB  192          ;ALØ
            DB    Ø          ;AL1
            DW   16          ;CKS
            DW    2          ;OFF

;HELYFOGLALÁS ÁTMENETI ADATOK SZÁMÁRA

DIRBUF:     DS             128
ALVØ:       DS              31
ALV1:       DS              31
CSVØ:       DS              16
CSV1:       DS              16
```

XII. FÜGGELÉK: BIOS MINTAPÉLDA

```
                ;
                ;BIOS MINTAPELDA
                ;

EA00                    ORG 0EA00H
D400 =          CPMB    EQU 0D400H
DC06 =          BDOS    EQU 0DC06H
1600 =          CPML    EQU X-CPMB
EA00 =          BIOS    EQU X
0000 =          BIAS    EQU 0           ;ELTOLAS
002C =          NSECTS  EQU CPML/128
0002 =          OFFSET  EQU 2           ;TRACK ELTOLAS
0004 =          NDISKS  EQU 4           ;DRIVE-OK SZAMA
0004 =          CDISK   EQU 4           ;ATMENETI TAROLO DRIVE KIJELÖLESHEZ
0080 =          BUFF    EQU 80H         ;BUFFER CIM
000A =          RETRY   EQU 10          ;HIBA ESETEN PROBALKOZAS SZAMA

                ;UGROTABLA

EA00 C3C2EA             JMP BOOT
EA03 C3D2EA     WBOOTE  JMP WBOOT
EA06 C3B2EA             JMP CONST
EA09 C3B5EA             JMP CONIN
EA0C C3BBEA             JMP CONOUT
EA0F C3BEEA             JMP LIST
EA12 C3BFEA             JMP PUNCH
EA15 C3C0EA             JMP READER

EA18 C362EB             JMP HOME
EA1B C367EB             JMP SELDSK
EA1E C37DEB             JMP SETTRK
EA21 C382EB             JMP SETSEC
EA24 C391EB             JMP SETDMA
EA27 C3A4EB             JMP READ
EA2A C3A7EB             JMP WRITE
EA2D C360EB             JMP LISTST
EA30 C387EB             JMP SECTRA
```

```
                ;4 DRIVE-OS RENDSZER
EA33 =          DPBASE  EQU X           ;DISZK PARAMETER BLOCK
EA33 82EA0000   DPE0:   DW XLT,0000H    ;TRANSZLACIOS TABLA
EA37 00000000           DW 0000,0000    ;ATMENETI BUFFER
EA3B AEEB73EA           DW DIRBUF,DPB   ;DIR.BUFF,PARAMETER BLOKK
EA3F 4CEC2EEC           DW CSV0,ALV0    ;ELLENORZESI,ALLOKACIOS VEKTOR
EA43 82EA0000   DPE1:   DW XLT,0000
EA47 00000000           DW 0000,0000
EA4B AEEB73EA           DW DIRBUF,DPB
EA4F 7AEC5CEC           DW CSV1,ALV1
EA53 82EA0000   DPE2:   DW XLT,0000
EA57 00000000           DW 0000,0000
EA5B AEEB73EA           DW DIRBUF,DPB
EA5F A8EC8AEC           DW CSV2,ALV2
EA63 82EA0000   DPE3:   DW XLT,0000
EA67 00000000           DW 0000,0000
EA6B AEEB73EA           DW DIRBUF,DPB
EA6F D6ECB8EC           DW CSV3,ALV3

EA73 =          DPB     EQU X           ;DISZK PARAMETER BLOCK
EA73 1A00               DW 26           ;SZEKTOR/TRACK
EA75 03                 DB 3            ;BLOCK ELTOLAS
EA76 07                 DB 7            ;BLOCK MASZK
EA77 00                 DB 0            ;EXTENT (KITERJESZTESI) MASZK
EA78 EF00               DW 239          ;DISZK MERET-1
EA7A 3F00               DW 63           ;DIRECTORY MAX
EA7C C0                 DB 192          ;ALLOC0
EA7D 00                 DB 0            ;ALLOC1
EA7E 1000               DW 16           ;ELLENORZO MERET
EA80 0200               DW 2            ;OFFSET
EA82 =          XLT     EQU X           ;TRANSZLACIOS TABLA
EA82 01070D1319         DB 1,7,13,19,25
EA87 050B111703         DB 5,11,17,23,3
EA8C 090F150208         DB 9,15,21,2,8
EA91 0E141A060C         DB 14,20,26,6,12
EA96 1218040A10         DB 18,24,4,10,16
EA9B 16                 DB 22
                ;
F01E =          MON     EQU 0F01EH      ;ROM-BA EGETETT MONITOR CIME

000D =          CR      EQU 0DH
000A =          LF      EQU 0AH

                SIGNON:
EA9C 0D0A0A             DB CR,LF,LF
EA9F 36304B2043         DB '60K CPM VERS X.X'
EAAF 0D0A00             DB CR,LF,0
                CONST:                  ;KONZOL STATUS
EAB2 C3E6EC             JMP SYS
                CONIN:                  ;KARAKTER BEOLVASAS AKKUBA
EAB5 CDE9EC             CALL SYS+3
EAB8 E67F               ANI  7FH        ;ASCII KARAKTER MASZK
EABA C9                 RET
```

```
                CONOUT:                 ;KARAKTER KIIRAS
EABB C3ECEC             JMP SYS+6

EABE 00         LIST:   NOP             ;NINCS ILYEN ESZKOZ A RENDSZERBEN
EABF 00         PUNCH:  NOP
EAC0 AF         READER: XRA A           ;AKKU NULLAZAS
EAC1 C9                 RET

                BOOT:
EAC2 310001             LXI SP,BUFF+80H
EAC5 019CEA             LXI B,SIGNON
EAC8 CD97EB             CALL PRMSG      ;SZOVEG KIIRAS A DISPLAY-RE
EACB AF                 XRA A
EACC 320400             STA CDISK       ;0.DRIVE KIVALASZTAS
EACF C31EEB             JMP GOCPM
EAD2 318000     WBOOT:  LXI SP,BUFF
EAD5 0E0A               MVI C,RETRY     ;OLVASASI KISERLETEK SZAMA
EAD7 C5                 PUSH B
EAD8 0100D4     WBOOT0: LXI B,CPMB
EADB CD91EB             CALL SETDMA
EADE 0E00               MVI C,0
EAE0 CD67EB             CALL SELDSK
EAE3 0E00               MVI C,0
EAE5 CD7DEB             CALL SETTRK
EAE8 0E01               MVI C,1
EAEA CD82EB             CALL SETSEC
EAED C1                 POP B
EAEE 062C               MVI B,NSECTS

EAF0 C5         RDSEC:  PUSH B          ;RENDSZER BEOLVASAS
EAF1 CDA4EB             CALL READ
EAF4 C248EB             JNZ BOOTER
EAF7 2AABEB             LHLD IOD
EAFA 118000             LXI D,128
EAFD 19                 DAD D
EAFE 44                 MOV B,H
EAFF 4D                 MOV C,L
EB00 CD91EB             CALL SETDMA
EB03 3AADEB             LDA IOS
EB06 FE1A               CPI 26          ;MAX 26 SZEKTOR/TRACK
EB08 DA14EB             JC RD1

EB0B 3AAAEB             LDA IOT
EB0E 3C                 INR A
EB0F 4F                 MOV C,A
EB10 CD7DEB             CALL SETTRK
EB13 AF                 XRA A
EB14 3C         RD1:    INR A
EB15 4F                 MOV C,A
EB16 CD82EB             CALL SETSEC
EB19 C1                 POP B
EB1A 05                 DCR B
EB1B C2F0EA             JNZ RDSEC
```

```
                GOCPM:
EB1E 018000             LXI B,BUFF
EB21 CD91EB             CALL SETDMA
EB24 3EC3               MVI A,JMP       ;0 LAP UGROCIMEINEK BEALLITASA
EB26 320000             STA 0
EB29 2103EA             LXI H,WBOOTE
EB2C 220100             SHLD 1
EB2F 320500             STA 5
EB32 2106DC             LXI H,BDOS
EB35 220600             SHLD 6
EB38 323800             STA 7*8
EB3B 211EF0             LXI H,MON       ;MONITOR BELEPESI CIMENEK BEIRASA
EB3E 223900             SHLD 7*8+1
EB41 3A0400             LDA CDISK
EB44 4F                 MOV C,A
EB45 C300D4             JMP CPMB        ;UGRAS A CCP-BE

                BOOTER:
EB48 C1                 POP B
EB49 0D                 DCR C
EB4A CA51EB             JZ BOOTE0
EB4D C5                 PUSH B
EB4E C3D8EA             JMP WBOOT0      ;UJABB BEOLVASASI KISERLET

                BOOTE0:
EB51 215AEB             LXI H,BOOTMS
EB54 CD97EB             CALL PRMSG
EB57 C31EF0             JMP MON

                BOOTMS:
EB5A 3F424F4F54         DB '?BOOT',0

                LISTST:                 ;LISTAZO ESZKOZ STATUS
EB60 AF                 XRA A
EB61 C9                 RET

                HOME:                   ;DISZK ALAPHELYZETBE ALLITAS
EB62 0E00               MVI C,0
EB64 C37DEB             JMP SETTRK

                SELDSK:                 ;DISZK KIVALASZTAS
EB67 210000             LXI H,0
EB6A 79                 MOV A,C
EB6B FE04               CPI NDISKS
EB6D D0                 RNC
EB6E 32AAEB             STA DBANK
EB71 69                 MOV L,C
EB72 2600               MVI H,0
EB74 29                 DAD H
EB75 29                 DAD H
EB76 29                 DAD H
EB77 29                 DAD H
EB78 1133EA             LXI D,DPBASE
EB7B 19                 DAD D           ;(HL)=A DRIVE-HEZ TARTOZO DPBASE
EB7C C9                 RET
```

```
                SETTRK:                 ;TRACK BEALLITAS
EB7D 21AAEB             LXI H,IOT
EB80 71                 MOV M,C
EB81 C9                 RET

                SETSEC:                 ;SZEKTOR BEALLITAS
EB82 21ADEB             LXI H,IOS
EB85 71                 MOV M,C
EB86 C9                 RET
                SECTRA:                 ;SZEKTOR TRANSZLACIO SZAMITAS
EB87 0600               MVI B,0
EB89 EB                 XCHG
EB8A 09                 DAD B
EB8B 7E                 MOV A,M
EB8C 32ADEB             STA IOS
EB8F 6F                 MOV L,A
EB90 C9                 RET
                SETDMA:                 ;DMA CIM BEALLITAS
EB91 69                 MOV L,C
EB92 60                 MOV H,B
EB93 22ABEB             SHLD IOD
EB96 C9                 RET

                PRMSG:                  ;SZOVEG KIIRASA. ZAROKARAKTER=0
EB97 7E                 MOV A,M
EB98 B7                 ORA A
EB99 C8                 RZ
EB9A E5                 PUSH H
EB9B 4F                 MOV C,A
EB9C CDBBEA             CALL CONOUT
EB9F E1                 POP H
EBA0 23                 INX H
EBA1 C397EB             JMP PRMSG

                READ:                   ;SZEKTOR BEOLVASAS
EBA4 C3EFEC             JMP SYS+9

                WRITE:                  ;SZEKTOR KIIRASA
EBA7 C3F2EC             JMP SYS+12

                ;ATMENETI TAROLO DRIVE PARAMETEREK SZAMARA

                DBANK:                  ;SZEKTOR KIIRASA        DB 0
EBAA 02         IOT:    DB OFFSET       ;TRACK
EBAB 8000       IOD:    DW BUFF         ;DMA CIM
EBAD 01         IOS:    DB 1            ;SZEKTOR
```

```
EBAE =          BEGDAT: EQU $
EBAE            DIRBUF: DS 128          ;DIRECTORY BUFFER
EC2E            ALV0:   DS 30
EC4C            CSV0:   DS 16
EC5C            ALV1:   DS 30
EC7A            CSV1:   DS 16
EC8A            ALV2:   DS 30
ECA8            CSV2:   DS 16
ECB8            ALV3:   DS 30
ECD6            CSV3:   DS 16
ECE6 =          ENDDAT  EQU $
0138 =          DATSIZ  EQU $-BEGDAT

                ;A BIOS HARDWARE SPECIFIKUS RUTINJAI
                ;A GEP MONITOR RUTINJAINAK HIVASA
                SYS:
ECE6 C300F0             JMP 0F000H
ECE9 C303F0             JMP 0F003H
ECEC C306F0             JMP 0F006H
ECEF C309F0             JMP 0F009H
ECF2 C30CF0             JMP 0F00CH

ECF5            END
```

XIII. FÜGGELÉK: COPY MINTAPROGRAM

COPY.ASM file

```
; FILE-TO-FILE COPY PROGRAM
;AT THE CCP LEVEL THE COMMAND
;       COPY A:X.Y B:U.V
;COPY THE FILE NAMED X.Y FROM DRIVE A
;TO A FILE NAMED U.V ON DRIVE B
BOOT    EQU 0 ;SYSTEM REBOOT
BDOS    EQU 0005H  ;BDOS ENTRY POINT
FCB1    EQU 005CH  ;FIRST FILE NAME
SFCB    EQU FCB1   ;SOURCE FCB
FCB2    EQU 006CH  ;SECOND FILE NAME
DBUFF   EQU 0080H  ;DEFAULT BUFFER
TPA     EQU 0100H  ;BEGINNING OF TPA
;
PRINTF  EQU 9      ;PRINT BUF FUNC#
OPENF   EQU 15     ;OPEN FILE FUNC#
CLOSEF  EQU 16     ;CLOSE FILE FUNC#
DELF    EQU 19     ;DELETE FILE FUNC#
READF   EQU 20     ;SEQUENTAL READ
WRITEF  EQU 21     ;SEQ WRITE
MAKEF   EQU 22     ;MAKE FILE FUNC#
;
        ORG  TPA          ;BEGINNING OF TPA
        LXI  SP,STACK.    ;LOCAL STACK
;MOVE SECOND FILE NAME TO DFCB
        MVI  C,16         ;HALF AN FCB
        LXI  D,FCB2       ;SOURCE OF MOVE
        LXI  H,DFCB       ;DESTINATION FCB
MFCB:   LDAX D            ;SOURCE FCB
        INX  D            ;READY NEXT
        MOV  M,A          ;DEST FCB
        INX  H            ;READY NEXT
        DCR  C            ;COUNT 16..0
        JNZ  MFCB         ;LOOP 16 TIMES
;
;NAME HAS BEEN MOVED, ZERO CR
        XRA  A            ;A=0
        STA  DFCBCR       ;CURRENT REC=0
;SOURCE AND DEST FCB'S READY
        LXI  D,SFCB       ;SOURCE FILE
        CALL OPEN         ;ERR IF 255
        LXI  D,NOFILE     ;READY MESSAGE
        INR  A            ;255 BECOMES 0
        CZ   FINIS        ;DONE IF NO FILE
```

```
;SOURCE FILE OPEN,PREP DESTINATION
        LXI   D,DFCB      ;DESTINATION
        CALL  DELETE      ;REMOVE IF PRESENT
;
        LXI   D,DFCB      ;DEST
        CALL  MAKE        ;CREATE THE FILE
        LXI   D,NODIR     ;READY MESSAGE
        INR   A           ;255 BECOMES 0
        CZ    FINIS       ;DONE IF NO DIR SPACE
;SOURCE FILE OPEN, DEST FILE OPEN.
;COPY UNTIL END OF FILE ON SOURCE
;
COPY:   LXI   D,SFCB      ;SOURCE
        CALL  READ        ;READ NEXT RECORD
        ORA   A           ;END OF FILE?
        JNZ   EOFILE      ;SKIP WRITE IF SO
;
;NOT END OF FILE, WRITE THE RECORD
        LXI   D,DFCB      ;DESTINATION
        CALL  WRITE       ;WRITE RECORD
        LXI   D,SPACE     ;READY MESSAGE
        ORA   A           ;00 IF WRITE OK
        CNZ   FINIS       ;END IF SO
        JMP   COPY        ;LOOP UNTIL EOF
EOFILE: ;END OF FILE, CLOSE DESTINATION
        LXI   D,DFCB      ;DESTINATION
        CALL  CLOSE       ;255 IF ERROR
        LXI   H,WRPROT    ;READY MESSAGE
        INR   A           ;255 BECOMES 0
        CZ    FINIS       ;SHOULDN'T HAPPEN
;
;COPY OPERATION COMPLETE,END
        LXI   D,NORMAL    ;READY MESSAGE
;
FINIS:  ;WRITE MESSAGE GIVEN BY DE, REBOOT
        MVI   C,PRINTF
        CALL  BDOS        ;WRITE MESSAGE
        JMP   BOOT        ;REBOOT SYSTEM
;SYSTEM INTERFACE SUBROUTINES
;(ALL RETURN DIRECTLY FROM BDOS)
OPEN:   MVI   C,OPENF
        JMP   BDOS
CLOSE:  MVI   C,CLOSEF
        JMP   BDOS
DELETE: MVI   C,DELF
        JMP   BDOS
READ:   MVI   C,READF
        JMP   BDOS
WRITE:  MVI   C,WRITEF
        JMP   BDOS
MAKE:   MVI   C,MAKEF
        JMP   BDOS
```

```
;CONSOLE MESSAGES
NOFILE: DB 'NO SOURCE FILES$'
NODIR:  DB 'NO DIRECTORY SPACES$'
SPACE:  DB 'OUT OF DATA SPACES$'
WRPROT: DB 'WRITE PROTECTED?$'
NORMAL: DB 'COPY COMPLETE$'
;
;DATA AREAS
DFCB:   DS 33           ;DESTINATION FCB
DFCBCR  EQU DFCB+32     ;CURRENT RECORD
;
        DS 32           ;16 LEVEL STACK
STACK:
        END
```

COPY.PRN file

```
                        ; FILE-TO-FILE COPY PROGRAM
                        ;AT THE CCP LEVEL THE COMMAND
                        ;       COPY A:X.Y B:U.V
                        ;COPY THE FILE NAMED X.Y FROM DRIVE A
                        ;TO A FILE NAMED U.V ON DRIVE B
0000 =                  BOOT    EQU 0 ;SYSTEM REBOOT
0005 =                  BDOS    EQU 0005H  ;BDOS ENTRY POINT
005C =                  FCB1    EQU 005CH  ;FIRST FILE NAME
005C =                  SFCB    EQU FCB1   ;SOURCE FCB
006C =                  FCB2    EQU 006CH  ;SECOND FILE NAME
0080 =                  DBUFF   EQU 0080H  ;DEFAULT BUFFER
0100 =                  TPA     EQU 0100H  ;BEGINNING OF TPA
                        ;
0009 =                  PRINTF  EQU 9      ;PRINT BUF FUNC#
000F =                  OPENF   EQU 15     ;OPEN FILE FUNC#
0010 =                  CLOSEF  EQU 16     ;CLOSE FILE FUNC#
0013 =                  DELF    EQU 19     ;DELETE FILE FUNC#
0014 =                  READF   EQU 20     ;SEQUENTAL READ
0015 =                  WRITEF  EQU 21     ;SEQ WRITE
0016 =                  MAKEF   EQU 22     ;MAKE FILE FUNC#
                        ;
0100                            ORG  TPA          ;BEGINNING OF TPA
0100 311E02                     LXI  SP,STACK     ;LOCAL STACK
                        ;MOVE SECOND FILE NAME TO DFCB
0103 0E10                       MVI  C,16         ;HALF AN FCB
0105 116C00                     LXI  D,FCB2       ;SOURCE OF MOVE
0108 21DD01                     LXI  H,DFCB       ;DESTINATION FCB
010B 1A                 MFCB:   LDAX D            ;SOURCE FCB
010C 13                         INX  D            ;READY NEXT
010D 77                         MOV  M,A          ;DEST FCB
010E 23                         INX  H            ;READY NEXT
010F 0D                         DCR  C            ;COUNT 16..0
0110 C20B01                     JNZ  MFCB         ;LOOP 16 TIMES
                        ;
                        ;NAME HAS BEEN MOVED, ZERO CR
0113 AF                         XRA  A            ;A=0
0114 32FD01                     STA  DFCBCR       ;CURRENT REC=0
                        ;SOURCE AND DEST FCB'S READY
0117 115C00                     LXI  D,SFCB       ;SOURCE FILE
011A CD6901                     CALL OPEN         ;ERR IF 255
011D 118701                     LXI  D,NOFILE     ;READY MESSAGE
0120 3C                         INR  A            ;255 BECOMES 0
0121 CC6101                     CZ   FINIS        ;DONE IF NO FILE
```

```
                        ;SOURCE FILE OPEN,PREP DESTINATION
0124 11DD01             	LXI   D,DFCB      ;DESTINATION
0127 CD7301             	CALL  DELETE      ;REMOVE IF PRESENT
                        ;
012A 11DD01             	LXI   D,DFCB      ;DEST
012D CD8201             	CALL  MAKE        ;CREATE THE FILE
0130 119701             	LXI   D,NODIR     ;READY MESSAGE
0133 3C                 	INR   A           ;255 BECOMES 0
0134 CC6101             	CZ    FINIS       ;DONE IF NO DIR SPACE
                        ;SOURCE FILE OPEN, DEST FILE OPEN
                        ;COPY UNTIL END OF FILE ON SOURCE
                        ;
0137 115C00             COPY:   LXI   D,SFCB      ;SOURCE
013A CD7801             	CALL  READ        ;READ NEXT RECORD
013D B7                 	ORA   A           ;END OF FILE?
013E C25101             	JNZ   EOFILE      ;SKIP WRITE IF SO
                        ;
                        ;NOT END OF FILE, WRITE THE RECORD
0141 11DD01             	LXI   D,DFCB      ;DESTINATION
0144 CD7D01             	CALL  WRITE       ;WRITE RECORD
0147 11AB01             	LXI   D,SPACE     ;READY MESSAGE
014A B7                 	ORA   A           ;00 IF WRITE OK
014B C46101             	CNZ   FINIS       ;END IF SO
014E C33701             	JMP   COPY        ;LOOP UNTIL EOF
                        EOFILE: ;END OF FILE, CLOSE DESTINATION
0151 11DD01             	LXI   D,DFCB      ;DESTINATION
0154 CD6E01             	CALL  CLOSE       ;255 IF ERROR
0157 21BE01             	LXI   H,WRPROT    ;READY MESSAGE
015A 3C                 	INR   A           ;255 BECOMES 0
015B CC6101             	CZ    FINIS       ;SHOULDN'T HAPPEN
                        ;
                        ;COPY OPERATION COMPLETE,END
015E 11CF01             	LXI   D,NORMAL    ;READY MESSAGE
                        ;
                        FINIS:  ;WRITE MESSAGE GIVEN BY DE, REBOOT
0161 0E09               	MVI   C,PRINTF
0163 CD0500             	CALL  BDOS        ;WRITE MESSAGE
0166 C30000             	JMP   BOOT        ;REBOOT SYSTEM
                        ;SYSTEM INTERFACE SUBROUTINES
                        ;(ALL RETURN DIRECTLY FROM BDOS)
0169 0E0F               OPEN:   MVI   C,OPENF
016B C30500             	JMP   BDOS
016E 0E10               CLOSE:  MVI   C,CLOSEF
0170 C30500             	JMP   BDOS
0173 0E13               DELETE: MVI   C,DELF
0175 C30500             	JMP   BDOS
0178 0E14               READ:   MVI   C,READF
017A C30500             	JMP   BDOS
017D 0E15               WRITE:  MVI   C,WRITEF
017F C30500             	JMP   BDOS
0182 0E16               MAKE:   MVI   C,MAKEF
0184 C30500             	JMP   BDOS
```

```
                ;CONSOLE MESSAGES
0187 4E4F20534FNOFILE: DB 'NO SOURCE FILES$'
0197 4E4F204449NODIR:  DB 'NO DIRECTORY SPACES$'
01AB 4F5554204FSPACE:  DB 'OUT OF DATA SPACES$'
01BE 5752495445WRPROT: DB 'WRITE PROTECTED?$'
01CF 434F505920NORMAL: DB 'COPY COMPLETE$'
                ;
                ;DATA AREAS
01DD            DFCB:   DS 33           ;DESTINATION FCB
01FD =          DFCBCR  EQU DFCB+32     ;CURRENT RECORD
                ;
01FE                    DS 32           ;16 LEVEL STACK
                STACK:
021E                    END
```

COPY.HEX file

```
:10010000311E020E10116C0021DD011A1377230D30
:10011000C20B01AF32FD01115C00CD6901118701F5
:100120003CCC610111DD01CD730111DD01CD8201F6
:1001300011970113CCC6101115C00CD7801B7C2512F
:10014000111DD01CD7D0111AB01B7C46101C337E0
:10015000111DD01CD6E0121BE013CCC610111CF49
:10016000010E09CD0500C300000E0FC305000E10DF
:10017000C305000E13C305000E14C305000E15C3FE
:1001800005000E16C305004E4F20534F55524345F0
:1001900020464494C4553244E4F2044495245435430
:1001A0004F525920535041434553244F5554204FEB
:1001B000462044415441205350414345532457521 3
:1001C000495445205052 4F5445435445443F2443DD
:0D01D0004F505920434F4D504C455445248D
:0000000000
```

COPY.SYM file

```
0005 BDOS      0000 BOOT      016E CLOSE     0010 CLOSEF    0137 COPY
0080 DBUFF     0173 DELETE    0013 DELF      01DD DFCB      01FD DFCBCR
0151 EOFILE    005C FCB1      006C FCB2      0161 FINIS     0016 MAKEF
0182 MAKE      010B MFCB      0197 NODIR     0187 NOFILE    01CF NORMAL
0169 OPEN      000F OPENF     0009 PRINTF    0014 READF     0178 READ
005C SFCB      01AB SPACE     021E STACK     0100 TPA       017D WRITE
0015 WRITEF    01BE WRPROT
```

XIV. FÜGGELÉK: GETSYS ÉS PUTSYS MINTAPROGRAMOK

```
;  GETSYS mintaprogram - beolvassa a Ø-ás és 1-es
;  sávot a memóriába a 338ØH cimtől kezdődően
:  Regiszterkiosztás:
;          A       átmeneti regiszter
;          B       sáv-számláló /Ø,1/
;          C       szektor-számláló /1,2,...,26/
;          DE      átmeneti regiszterpár
;          HL      memóriacim
;          SP      stack pointer
START:
          LXI   SP,338ØH    ;SP=stack teteje
          LXI   H,338ØH     ;HL=1.memóriacim
          MVI   B,Ø         ;sáv=Ø
RDTRK:
          MVI   C,1         ;1-es szektor olvasása
RDSEC:
          CALL  READSEC     ;felhasználó által megirt
                            ;szubrutin egy szektor le-
                            ;olvasására
          LXI   D,128
          DAD   D           ;memóriacim növelése
                            ;128-cal
```

```
        INR     C               ;szektor=szektor+1
        MOV     A,C             ;sáv vége?
        CPI     27
        JC      RDSEC           ;ha nem, C=1, ugrás a
                                ;következő szektor beolvasá-
                                ;sára
        INR     B               ;sáv-számláló növelése
        MOV     A,B
        CPI     2               ;utolsó /2./ sáv?
        JC      RDTRK           ;ha nem, ugrás a következő
                                ;/2./ sáv beolvasására
        HLT                     ;vége a GETSYS szubrutinnak


;   PUTSYS mintaprogram - kimenti a memória tartalmát
;   a 3380H cimtől kezdődően a lemez Ø-ás és 1-es
;   sávjára
;   Regiszterkiosztás:
;       A       átmeneti regiszter
;       B       sáv-számláló /Ø,1/
;       C       szektor-számláló /1,2,...,26/
;       DE      átmeneti regiszterpár
;       HL      memóriacim
;       SP      stack pointer
START:
        LXI   SP,3380H          ;SP=stack teteje
```

```
        LXI   H,3380H       ;HL=1.memóriacim
        MVI   B,Ø           ;sáv=Ø
WRTRK:
        MVI   C,1           ;szektor=1
WRSEC:
        CALL  WRITESEG      ;felhasználó által megirt
                            ;szubrutin egy szektor ki-
                            ;irására
        LXI   D,128
        DAD   D             ;memóriacim növelése 128-cal
        INR   C             ;szektor=szektor+1
        MOV   A,C           ;sáv vége?
        CPI   27
        JC    WRSEC         ;ha nem, ugrás a következő
                            ;szektor kiirására
        INR   B             ;különben a sáv-számláló
                            ;növelése
        MOV   A,B
        CPI   2             ;utolsó /2./ sáv?
        JC    WRTRK         ;ha nem, ugrás a következő
                            ;/2./ sáv kiirására
        HLT                 ;egyébként vége a PUTSYS
                            ;szubrutinnak
```

A kézikönyvben alkalmazott jelölések

Az alábbiakban összefoglaltuk azokat a jelölési konvenciókat, melyeket a kézikönyvben végig következetesen alkalmaztunk. Ezek egy részére a nyomdai jelek összecserélhetősége miatt volt szükség, más részüket az indokolja, hogy a különböző gépek különböző klaviaturával és a billentyükön eltérő felirattal rendelkeznek; ennek ellenére szükséges, hogy ugyanazt a funkciót mindig ugyanugy jelöljük. A jelölések további része lényeges tartalmi jelentéssel bir.

A nulla és az O-betü megkülönböztetése céljából a nulla szám jelölése a szövegben mindenütt $\emptyset$, mig a betü mindenhol áthuzás nélkül szerepel: O.

A kocsi vissza billentyü /szokásos felirata a különböző géptipusokon: RETURN, ENTER, CR, stb./ jele a kézikönyvben, ha szövegben szerepel: CR
, ha példában szerepel: ↵

Az ugynevezett vezérlő-, vagy kontroll karakterek, melyekkel képernyőszerkesztési funkciók valósithatók meg, két billentyü egyidejü lenyomását igénylik: az egyik az általában "Control" felirattal rendelkező vezérlő-billentyü, a másik az adott funkciónak megfelelő betü. A kontrollkarakterek jelölése a szövegben a CTRL betücsoport és mellette kötőjel után a megfelelő nagybetü, pl.: CTRL-C.

A számok végén a számmal egybeirt H jelentése, hogy a szám hexadecimális számrendszerben értendő, mig a D a decimális számokat jelöli /ez utóbbit csak ott tüntettük fel, ahol az egyértelmüség követelménye megkivánta/.

A példákban folyamatos vonallal huztuk alá azokat a szövegrészeket, melyeket a felhasználónak kell bebillentyüznie.

Szaggatott vonallal huztuk alá a példákban azokat a részeket, melyek megadása lehetséges, de nem kötelező /opcionális paraméterek/.

A kézikönyvben szereplő rövidítések jegyzéke

ACC:	Access	-hozzáférés/i mód/
ASCII:	American Standard Code for Information Interchange	
BDOS:	Basic Disc Operating System	-alapvető lemezkezelő rendszer
BIOS:	Basic Input Output System	-alap-periféria kezelő rendszer
BYTS:	Bytes	-byte-ok
CCP:	Console Command Processor	-konzol-parancs feldolgozó
CMUG:	CP/M User's Group	-CP/M felhasználók köre
CPU	Central Processing Unit	-központi egység
CR:	Carriage Return	-kocsi-vissza
CRLF:	Carriage Return - Line Feed	-kocsi-vissza soremelés
CRT:	Cathode Ray Tube	-katódsugárcső /képernyő/
CTRL:	CONTROL	-a klaviatura vezérlőbillentyüje
C64:	Commodore 64	
DDT:	Dynamic Debugging Tool	-Dinamikus hibafelderitő program
DIR:	Directory	-/lemez/nyilvántartó
DMA:	Direct Memory Access	-közvetlen memória-hozzáférés
DPB:	Disc Parameter Block	-lemez paraméterblokk
DPT:	Disc Parameter Table	-lemez paraméter táblázat
ED:	Editor	-szövegszerkesztő program
EOF:	End of File	-file vége

EXT:	Extent	-file-terület v.blokk
FCB:	File Control Block	-file-leiró blokk
FDOS:	Floppy Disc Operating System	-hajlékony lemez kezelő rendszer
I/O :	Input/Output	-bemeneti/kimeneti
K:	Kilobyte	-1ø24 byte
LF:	Line feed	-soremelés
PIP:	Peripheral Interchange Program	-perifériák közötti adatátvitelt megvalósitó program
RAM:	Random Access Memory	-közvetlen hozzáférésü,irható-olvasható memória
RECS:	Records	-rekordok
R/O:	Read only	-csak olvasható /file/
R/W:	Read/Write	-irható/olvasható /file/
SID:	Symbolic Image Debugger	-szimbólikus hibafelderitő program
TPA:	Transient Program Area	-tranziens programterület
Z8ø:	Zilog 8ø	

I N D E X

ACC 44, 45
akkumulátor 13Ø, 131, 2Ø5
állapotregiszterek 143, 147
állapotvektor 25Ø
argumentum 26, 54
ASCII 27, 29, 68
ASCII-file 31, 35, 68, 77
assembler 2, 7, 1Ø6
assemblálás 113, 12Ø, 16Ø
assebly-nyelvü programozás 32, 1Ø5, 1Ø6
ASM 2, 29, 59, 1Ø6, 16Ø
átmeneti file 29, 89, 92
attributum 46, 5Ø

backup 55, 86
.BAK 29, 86
balról-jobbra szabály 121
.BAS 28, 29
BASIC 3, 7, 29
BAT: 48, 71, 2Ø2
batch-feldolgozás 32
BATCH-üzemmód 2Ø2
báziscim 149
BDOS 2, 18, 192, 193
belépési pont 19
bemeneti-kimeneti perifé-ria 68
BIOS 18, 19, 192
BIOS-szubrutinok 2Ø4, 212
BIOS-ugrótábla 199, 224, 313
bit 119,
blokk 51, 216
blokkos átvitel 73
BOOT 198, 2ØØ
bootstrap loader /kezdeti be-töltő/ 9, 198
bus 1Ø
byte 15
BYTS 44

C 7
C 64 1O, 11
CALL 2Ø, 131
Carry 132, 135
Carry-bit /átvitel-bit/ 13Ø, 153
cartridge 1Ø, 11
CBIOS 264, 267
CBOOT 27Ø, 271
CCP 18, 3Ø, 192
Cél-lemezegység 4Ø
checksum /ellenőrző összeg/ 76
cimke 112, 113
cimmutató 129
cimzési mód 7
CMUG 33
.COB 29
COBOL 29
.COM 27, 29, 32

Commodore-64 1Ø, 34, 44
compiler 7
CON: 48, 69, 2ØØ
COPY 11, 67, 269, 322
CP 92
CPU 6, 15
CPU-regiszterek 2Ø
CP/Net 3
CR 11, 26
.CRF 29
CRT: 47, 1Ø, 2Ø2
csoportos file-hivatkozás 28

.DAT 29
DB 122, 126
DDT 2, 59, 14Ø, 163, 3Ø9
debugger 2, 163
decimális carry-bit 153
DESTINATION 4Ø, 273
Digital Research 23, 14Ø
DIR 11, 31, 33, 46, 164
directory 3Ø, 196
directory bejegyzés 3Ø, 196
directory ellenőrzési vektor 219
directory entry 3Ø, 5Ø, 232
directory kód 242
directory puffer 214
direktiva 111, 121
disassembler 272
diskette 21
display 6, 68
DMA 2Ø8, 251
DMA-cim 2Ø3, 213, 24Ø, 251
DMA-puffer 21Ø, 244
DPB 214, 215, 22Ø
DPT 2Ø7, 213
DPT-fejléc 213, 214, 215
drive 39
driver-rutin 19, 47, 192
DS 122, 128
DSK 49, 5Ø
DUMP 32, 6Ø, 156, 166
DW 122, 127
Dynamic Debugging Tool 59, 1Ø6, 14Ø,

echózás 74
ED 2, 58, 85, 167
editálás 25, 85, 167
editor 2, 58, 85
egyedi file-hivatkozás 28, 31
egyszeres pontosságu 132
ellenőrző összeg 76
ellenőrzött file-név 222
előjel-bit 131
elsőszintü rendszergenerálás 263, 266
emlékeztető-név 7
END 113, 122, 123
END IF 122, 125
end-of-file 71, 77
EOF 71
EQU 114, 122, 124

ERA 26, 31, 36, 168
ERASE 31, 36
eredmény-file 86, 1Ø7
EXT 44

FCB 2Ø, 195, 229, 231
FDOS 193
felhasználói terület 67,81
felhasználó-kód 258
feltételes assemblálás 122
file 18, 27
file-allokáció 3Ø
file-attributum 5Ø, 82
file-hivatkozás 28
file-kezelés 7, 27, 193
file-kezelő 2, 27
file-leiró tábla 2Ø
file-méret 42
file-név 27
file-terület 3Ø, 51, 216
file-tipus 27, 28
file-törlése 3Ø, 245
fizikai egység név 47, 69
fizikai periféria 47
fizikai szektorszám 21Ø
flag 222
floppy-disc 2, 21
.FOR 29
forditóprogram 2, 7
FORMAT 23
formattálás 23
form-feed 75, 78
forrás-file 85
forráskönyvtár 92, 99
forrásnyelvi file 1Ø7, 139
forrás-sor 9Ø
FORTH 7
FORTRAN 7, 29
funkcióbillentyü 11, 12
funkció-sorszám 19, 226

gépi kód 7, 59, 6Ø, 1Ø5
GETSYS 265, 267, 329

hajlékony lemez 2, 6, 2Ø
hardware 2, 8
hardware-konfiguráció 2, 197
háttértároló 2, 27, 211
.HEX 29, 6Ø, 1Ø7
hexadecimális 15, 115
hideginditás 9, 267
host-mikrogép 61, 171

IF 122, 125,
indikátor 49, 5Ø
információs vektor 252
INP: 72, 289
input-file 86, 1Ø9
input-output 2, 46, 68
input-output program 19
.INT 29
Intel 1Ø, 1Ø5
interface 1Ø, 19
interpreter 3, 7

I/O 1Ø, 18, 46
IOBYTE 49, 2Ø1
I/O processzor 1Ø
irássürüség 22, 23
irásvédelem 22, 253
iró-olvasó fej 21

joker-karakter 28, 66, 245

K 15
karakter-mutató 91
karakter-pointer 55, 91, 94
karaktersorozat 19, 85, 118
kártyalyukasztó 69, 7Ø
kártyaolvasó 69, 7Ø
kazettás adatrögzitó 7Ø, 73
Kbyte 11, 15
képernyő-egység 47
képernyős kijelző 6
keresztreferencia 29
kétszeres pontosságu 132
kezdeti betöltő 9, 198
kezdőcim 122
kiterjesztés 27, 1Ø9, 227
kommunikációs regiszter 19, 226
komplementálás 137
konfiguráció 2, 18
konkatenáció 63, 67
konzol 19, 2Ø, 25, 48
konzolparancs-feldolgozó 17, 3Ø
központi egység 6
központi tár 6,15
kurrens felhasználói terület 75, 83
kurrens lemezegység 24, 34, 159.
kurzor 25

lemezegység 9, 21
lemezegység-azonositó 35, 2Ø7
lemezes file 2Ø, 27, 42
lemez-vezérlő 2, 221
lemez-vezérlő egység 6, 197
lemez-vezérlő interface 19
léptető faktor 21Ø
LF 74
.LIB 92, 99
LOAD 11, 32, 6Ø, 154, 17Ø
logikai egység-név 46, 69
logikai egységszám 221
logikai inverz 119
logikai periféria 46, 48, 2Ø1
logikai szektorszám 21Ø
LPT: 47, 4Ø, 2Ø3
.LST 29
LST: 47, 69, 2Ø1
lyukszalag 2

MØ8X 19
.MAC 29
magasszintü nyelv 7
mainframe 5

makró 138
makroassembler 213
második szintü rendszergenerálás 265, 269
megszakitás 137
melegindítás 9, 17, 2Ø4
memória 9, 15
memóriablokk 15, 148
memóriacim 15, 37
memóriafelosztás 15, 211
memória-lap 37, 211
memória-lapszám 37
memóriaméret 17, 9Ø, 1Ø2
memória-puffer 89, 2Ø3
merev lemezes tároló 2Ø
mikroprocesszor 4, 1Ø5, 137
mikroszámitógép 2, 5
mini-diskette 21
miniszámitógép 5
minusz-bit 153
mnemonik 7, 129
modem 6, 47
MOVCPM 18, 61, 171, 269
MP/M 3
müveleti kód 12Ø

nagyszámitógép 5
NEXT 37, 15Ø, 179
NUL: 71, 289
numerikus konstans 114
nyelvi forditók 33
nyelvi rendszer 8, 28
nyelvi szerkesztők 8
nyomkövetés 142

on-line 241, 25Ø
operációs rendszer 2, 4, 7
operandus 111, 113, 117, 118
operandusmező 112, 118
ORG 122
OUT: 72, 29Ø
output-file 86, 1Ø9, 139
overflow 138
overlapping 123
.OVR 29

papirszalag-lyukasztó 47, 7Ø, 75
papirszalag-olvasó 47, 7Ø, 75
paraméterblokk 2Ø, 214
parancs-mód 18
parancs-sor 27, 42
parancs-sorozat 89, 1ØØ
paritás-bit 72, 3Ø
PASCAL 7
patching 72
pariféria 6, 18, 69, 192
periféria-kezelés 19, 197
periféria-vezérlő 6
Peripheral Interchange Program 63, 2Ø1
PIP 29, 56, 63, 173
PIP-paraméterek 65, 73, 174, 29Ø

PL/M 7
plotter 6
printer 1Ø, 19
prioritási sorrend 121
.PRN 29, 1Ø9
PRN: 72, 289
processzor 68
program-mód 18
programozási nyelv 7
programszámláló /PC/ 135, 146, 153
programszegmens 142
prompt 11, 24, 26
Proper-8 12
PROPOS 12, 13
pszeudo-utasitás 112, 121
PTP: 47, 7Ø
PTR: 47, 7Ø
puffer 55, 89, 194, 2Ø8
pufferelt input 54, 55
PUN: 47, 69, 2Ø1
PUTSYS 265, 266, 329

RAM 6, 198
RDR: 47, 69, 2Ø1
Read-only 63, 241, 252
RECS 44
regiszter 19, 116
regiszter-pár 129, 134
rekonfigurálás 61, 171
rekord 43, 5Ø
.REL 29
REN 32, 35, 178
rendszer-file 46, 63, 79
rendszerfunkcióju szubrutinok 232
rendszerfunkciók 19, 226
rendszergenerálás 32, 264
rendszer-hivás 19, 226
rendszer-inicializálás 198, 232
rendszerlemez 11, 24
rendszerparaméterek 211
rendszer-prompt 26
rendszer/segéd/programok 6, 8, 3Ø
reset 9, 55, 56
RETURN 25
rezidens parancs 18, 32, 33, 159, 194
R/O 43, 45, 46, 63, 8Ø
ROM 6, 198
RUBOUT 53, 93, 141
RUN 11
R/W 45, 46

sáv 9, 21
sáv-cim 2Ø7
SAVE 32, 37, 179
SET 114, 122, 125
SID 212, 3Ø8
software 2, 6, 8
software ellenőrzési terület 214, 219
sornyomtató 6, 26, 69
sor-orientált 58, 74

SOURCE 39, 186
source-file 87
space 89
stack 135
Stack Pointer /SP/ 135, 143, 153
STAT 11, 4Ø, 18Ø
string 78, 79, 85
.SUB 29, 51, 183, 189
SUBMIT 29, 51, 183, 189
.SYM 29, 328
.SYS 46, 79, 82, 256
SYSGEN 18, 39, 185, 265
.$$$ 29

számkonstans 113, 115
szektor 9, 21
szektorméret 21, 51
személyi számitógép 4, 5
szimbólumtábla 29
szöveg-file 35, 58, 85
szöveg-konstans 118, 119
szövegszerkesztő 3, 8, 58, 85
szubrutin 72, 138
szubrutin-hivás 13Ø

TAB-karakter 72, 79
tabulátor 79, 145
tabulátor-pozició 79
tárgykód 77
teletype-egység 47, 2Ø2
temporary file 89
térkép 214
terminál 6, 48
text-file 58, 85
töréspont 141, 146
TPA 16, 38, 123, 154
trace 142, 151
track 21, 51
tranziens parancs 8, 3Ø, 31, 39
tranziens program 3Ø, 31, 194, 211, 212
tranziens programterület 16, 195
TTY: 47, 7Ø, 2Ø2
.TXT 34
TYPE 31, 35, 187, 194

UC1: 47, 7Ø, 2Ø2
ugrótábla 17, 199
ujrakonfigurálás 61
UL1: 47, 71, 2Ø3
untrace 142, 153
UP1: 47, 71, 2Ø3
UP2: 47, 71, 2Ø3
UR1: 47, 7Ø, 2Ø2
UR2: 47, 71, 2Ø2
USER 81, 83, 188
USR: 49
utility 6, 8, 3Ø1

végcim 15
véletlen elérésü lemezkezelés 18

verziószám 239
vezérlőkarakterek 25, 141, 294
video-monitor 6, 1Ø
video-terminál 6, 68, 79
virtuális file-méret 258

WBOOT 2ØØ, 2Ø4
Winchester-lemez 6, 21
word-processor 58

.XRF 29
XSUG 34, 54, 189

Z8Ø 1Ø
Zéró-bit 13Ø, 153

Ára: 250,– Ft